Silvano Sotelo Pião

IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL EM FPGA UTILIZANDO VHDL

Trabalho de Conclusão de Curso apresentado à
Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em
Eletrônica

ORIENTADORA: Luiza Maria Romeiro Codá

São Carlos
2012

Ficha catalográfica preparada pela Seção de Tratamento

da Informação do Serviço de Biblioteca – EESC/USP

Pião, Silvano Sotelo
P581i Implementação de rede neural artificial em FPGA utilizando VHDL / Silvano Sotelo Pião; orientadora Luiza Maria Romeiro Codá. São Carlos, 2012.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. redes neurais artificiais. 2. Field Programmable Logic Devices. 3. VHSIC Hardware Description Language. 4. dispositivos lógicos. 5. Perceptron. 6. Cyclone IV. I. Título.

FOLHA DE APROVAÇÃO

Nome: Silvano Sotelo Pião

Título: "Implementação de Rede Neural Artificial em FPGA utilizando VHDL"

Trabalho de Conclusão de Curso defendido e aprovado em 27 / 11 / 2012,

com NOTA 10,0 (DEZ , ZERO), pela Comissão Julgadora:

Profa. Assistente Luiza Maria Romeiro Codá (Orientadora)
SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva
SEL/EESC/USP

Prof. Dr. Danilo Hernane Spatti
SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Dedicatória

Dedico este trabalho aos meus pais, Katia S. Sotelo Pião e Silvano Pião, que me ensinaram com o exemplo de suas próprias atitudes a trilhar os caminhos certos, e que por sua dedicação incondicional aos estudos de seus filhos, permitiram a mim e minhas irmãs nos tornarmos quem somos hoje.

Agradecimentos

Agradeço à minha família, que sempre unida, me deu a base necessária para ingressar nesta Universidade e o apoio para superar todos os desafios.

À minha orientadora Luiza Maria Romeiro Codá, que pelas atividades que realizamos juntos durante os últimos anos da minha graduação, teve um papel fundamental na execução deste trabalho, e no meu crescimento acadêmico e profissional.

Aos professores Ivan Nunes da Silva e Danilo Hernane Spatti pelos valiosos conselhos oferecidos.

Resumo

Embora o estudo de redes neurais tenha se iniciado há quase sete décadas, esta área ainda apresenta uma imensa capacidade de evolução. A maioria das redes neurais existentes atualmente opera em máquinas seqüenciais, devido principalmente, ao menor custo quando comparadas às alternativas. No entanto, tais máquinas são incapazes de reproduzir uma característica inerente a redes neurais biológicas: a execução de operações de adição e multiplicação em paralelo. Com as evoluções observadas na última década em FPGAs *(Field Programmable Logic Devices)*, tanto na redução dos preços, quanto no aumento da capacidade, estes constituem uma excelente alternativa à implementação de redes neurais com execução de múltiplas tarefas em paralelo ao nível de *hardware*. Os FPGAs atuais possuem densidade de elementos lógicos suficientes para implementar grandes redes neurais. A utilização de *VHSIC Hardware Description Language* proporciona a possibilidade de intercambiabilidade entre dispositivos e facilita a alteração de características do circuito. Com o objetivo de estudar esta alternativa, este trabalho apresenta a síntese de uma rede *Perceptron* de duas entradas em um FPGA Cyclone IV EP4CE115F29C7N, da Altera, e o estudo de variações da rede sintetizada e suas implicações no consumo de elementos lógicos do dispositivo.

Palavras-Chave: redes neurais artificiais, *Field Programmable Logic Devices*, *VHSIC Hardware Description Language*, dispositivos lógicos, Perceptron, Cyclone IV.

Abstract

Although the study of neural networks has begun almost seven decades ago, this area still has an immense capacity for evolution. Most existing neural networks currently operate in sequential machines, mainly, due to lower cost compared to alternatives. However, such machines are unable to reproduce a feature of biological neural networks: the execution of addition and multiplication operations in parallel. With the developments observed in the last decade in FPGAs (Field Programmable Logic Devices), both in reducing prices, as the increased capacity, these are a great alternative to the implementation of neural networks with multiple tasks running in parallel in hardware level. The current FPGAs have sufficient density of logic elements for implementing large neural networks. The use of VHSIC Hardware Description Language provides the possibility of interchangeability of devices and makes easier changing characteristics of the circuit. With the aim of studying this alternative, this work presents the synthesis of a *Perceptron* network with two inputs on a Cyclone IV FPGA EP4CE115F29C7N, by Altera, and the study of variations of the network synthesized and its implications in the consumption of logic elements of the device.

Keywords: artificial neural networks, Field Programmable Logic Devices, *VHSIC Hardware Description Language*, logic devices, Perceptron, Cyclone IV.

Lista de Figuras

Lista de Tabelas

Sumário

1. Introdução

O funcionamento do cérebro é objeto de fascínio e curiosidade desde que começou a ser estudado. Sua capacidade, até hoje inigualada, de aprendizado, memória e adaptação tem levado cientistas a tentar reproduzi-lo há quase sete décadas (Haykin S. S., 1999).

Os primeiros estudos registrados sobre (RNAs) Redes Neurais Artificiais datam de 1943, em artigos de Warren McCulloch e Walter Pitts, respectivamente médico e estatístico, quando foi construído um circuito elétrico de uma rede neural simples, na tentativa de entender melhor o funcionamento do cérebro humano (Kovács L. Z., 2006).

As RNAs passaram a ser fortemente estudadas somente no início dos anos 90 e, embora sejam aplicadas em diferentes áreas, ainda apresentam um potencial de pesquisa imenso (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

A recente evolução dos FPGAs (*Field Programmable Gate Arrays*) tem apontado novos rumos para o desenvolvimento de RNAs, uma vez que permitem a implementação de redes com execução de tarefas em paralelo, característica semelhante ao funcionamento do cérebro biológico, impossível de se reproduzir por implementação em *softwares* para processadores seqüenciais (Hariprasath S., Prabakar T. N., 2012).

1.1. Motivação

RNAs são constituídas por unidades de processamento individuais – baseadas nos neurônios – que se comunicam por sinapses com pesos. A rede neural é capaz de aprender baseada em exemplos, permitindo que uma tarefa seja realizada sem conhecimento de regras pré-definidas. A generalidade deste conceito faz com que as RNAs sejam adequadas para uma ampla área de aplicações, como reconhecimento de voz, reconhecimento óptico de caracteres, prospecção de dados, previsões do mercado financeiro e diagnósticos médicos (Domingos P. O., Silva F. M., Neto H. C., 2005). Podem ser utilizadas para controlar sistemas físicos complexos e não lineares com parâmetros desconhecidos. Têm sido aplicadas com sucesso em controle de braços robóticos, controle de processos químicos, produção em massa de peças de alta qualidade e aplicações aeroespaciais.

Como a maioria dos *softwares* de redes neurais é executada por arquiteturas seqüenciais (Coric S., Latinovic I., Pavasovic A., 2000), não é possível realizar múltiplas operações de soma e subtração, que são características típicas de redes neurais. Somente a implementação em hardware de unidades capazes de realizar operações de soma e multiplicação, que operem em paralelo, é capaz de produzir redes rápidas. Logo após o ressurgimento das pesquisas sobre redes neurais em meados dos anos 80, percebeu-se que para um total proveito das capacidades de aprendizado e paralelismo massivo, destas, é essencial a implementação em *hardware* (Hariprasath S., Prabakar T. N., 2012).

Os FPGAs são uma excelente opção para a implementação de RNAs, pois permitem a integração de milhares de portas lógicas em um único dispositivo. As Diferentes opções de configuração destes dispositivos oferecidas por *softwares* específicos, como o Quartus II da Altera – que permite criação de circuitos com diagrama de blocos, VHDL ou Verilog – facilitam o projeto dos circuitos. Além disso, a evolução da Lei de Moore permite que se observem avanços em tecnologias FPGA em termos de potência, desempenho e preço (Bélanger N. L., 2007), como ilustrado pela Figura 1.

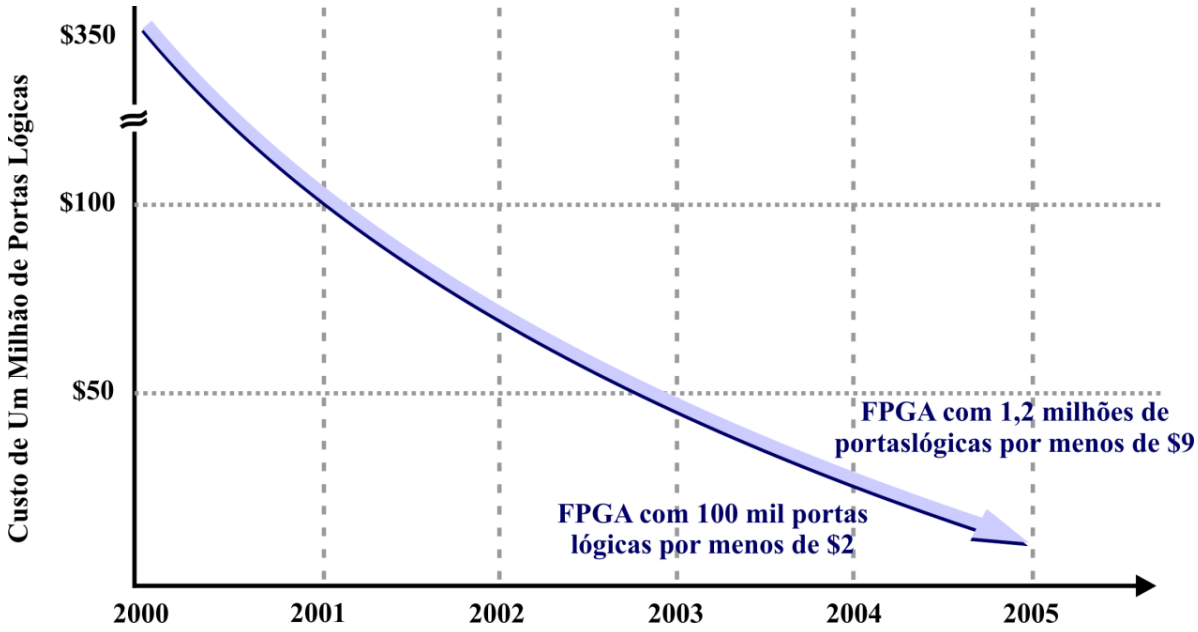


Figura 1 - Evolução no custo de FPGAs

Fundamentada na imagem publicada em: http://dsp-fpga.com/article-id/?2095

Um estudo baseado no "*International Technology Roadmap for Semiconductors"* exibiu a imagem da Figura 2, com a expectativa de evolução na capacidade de FPGAs. A constante evolução destes dispositivos mostra que projetistas de circuitos podem escolher FPGAs como uma prova de conceito, ou mesmo como produto final (Nieto A., Brea V. M., Vilariño D. L., 2009).

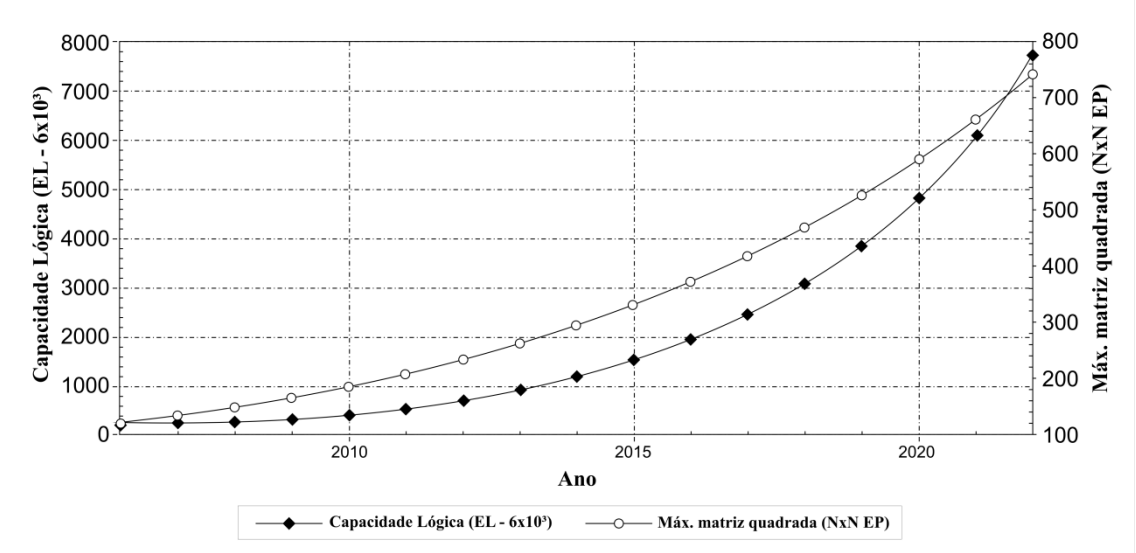


Figura 2 – Estimativa de evolução na capacidade de FPGAs

Fundamentada na imagem publicada em: http://www.intechopen.com/books/image-processing/an-fpga-based-topographic-computer-for-binary-image-processing

Entre as diversas opções para a configuração de um circuito em um FPGA, a VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) apresenta-se como uma excelente alternativa. Tendo sua última atualização em 1993, é uma linguagem relativamente recente e, portanto, apresenta características que facilitam a síntese de circuitos, como a possibilidade de parametrização em função de constantes, que podem ser definidas no momento da síntese, e o suporte a projetos com múltiplos níveis de hierarquia, este último é particularmente interessante à criação de RNAs. A semelhança desta linguagem com linguagens de programações amplamente utilizadas, como C a tornam de fácil aprendizado. Padronizada pelo IEEE (*Institute of Electrical and Electronics Engineers*), a VHDL possibilita o intercâmbio de informações referentes ao comportamento de um circuito entre fabricantes, fornecedores de sistemas e empresas de projetos (d'Amore R., 2005).

Com a linguagem VHDL pode-se, portanto, descrever e sintetizar um único neurônio, isoladamente, e testá-lo até que se obtenha o resultado desejado, independente do modelo de FPGA que será utilizado. O neurônio obtido pode então ser inserido em um circuito maior, quantas vezes forem necessárias até que se obtenha a rede neural desejada, ou pode, até mesmo, ser utilizado em diversos projetos distintos, em quantidades distintas, formando diferentes redes neurais.

Com a descrição do circuito da rede neural concluída, pode-se verificar se o FPGA desejado possui componentes internos suficientes para sua síntese. Em caso negativo, é possível averiguar os requisitos informados pelo software, utilizado na descrição do hardware, para encontrar um dispositivo adequado. Pode-se, também, buscar possíveis alterações no projeto

com o objetivo de reduzir os recursos exigidos. Esta característica facilita a escolha de um dispositivo ideal ao projeto, considerando capacidade e custo, fatores fundamentais para um projeto de engenharia. A Figura 3 exibe um exemplo de descrição de hardware evidenciando o número de componentes internos exigidos para o projeto. Embora a imagem exibida seja de um projeto realizado no Quartus II da Altera, a possibilidade de verificar tais requisitos não é exclusiva deste, podendo-se aplicar a mesma técnica a dispositivos de outros fabricantes.

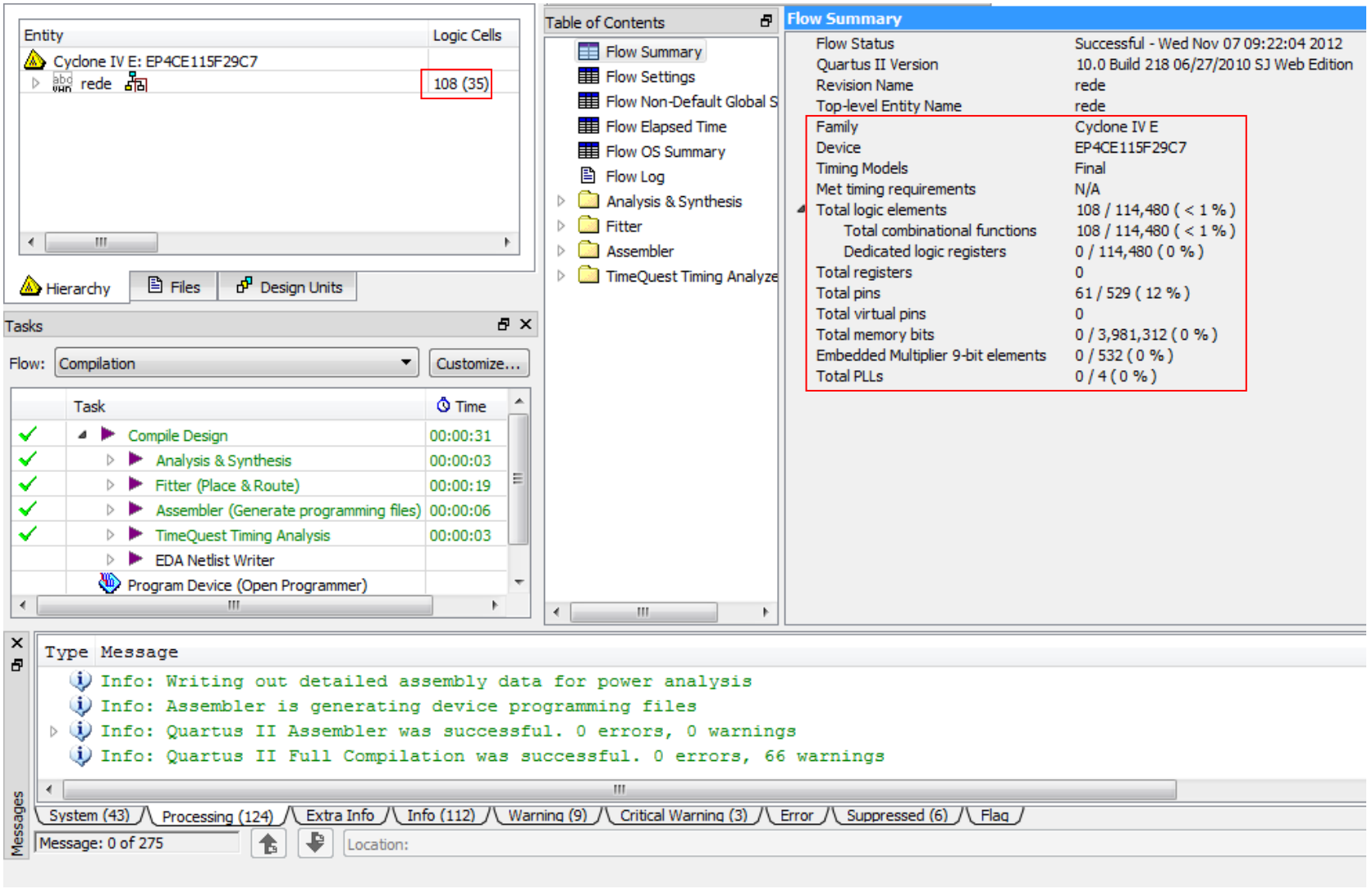


Figura 3 – Interface do *software* Quartus II, exibindo o número de componentes internos do FPGA utilizados no projeto.

1.2. Objetivo

Este trabalho tem como objetivo sintetizar uma rede do tipo *Perceptron* de duas entradas em um FPGA utilizando VHDL para descrição do circuito lógico. A validação do funcionamento da rede é possível pela comparação dos valores obtidos na saída com os valores teóricos de uma rede de referência. O número de elementos lógicos exigidos do FPGA é comparado a diferentes sínteses da mesma rede, com variações na precisão da representação numérica.

O FPGA utilizado é o Cyclone IV EP4CE115 da Altera e está inserido no *kit* de desenvolvimento DE2-115 da Terasic (Terasic Technologies Inc.). Os valores de entrada são informados à rede através de *slide switches*, e o valor de saída é exibido por um *LED (Light-Emitting Diode)* e por *displays* de sete segmentos, através de um decodificador sintetizado no próprio FPGA. A Figura 4 exibe o *kit* de desenvolvimento DE2-115 com indicação de seus componentes.

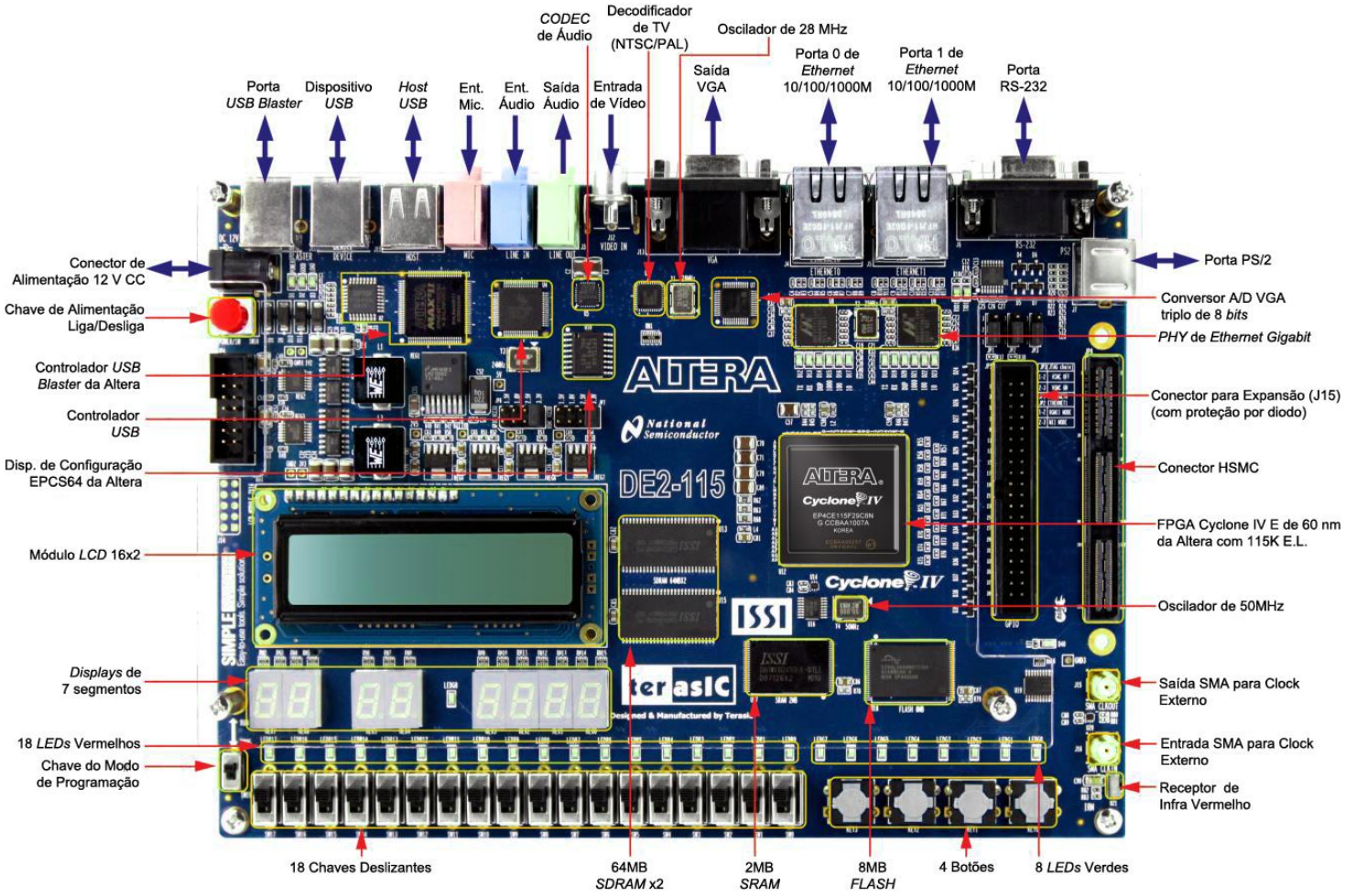


Figura 4 – *Kit* de desenvolvimento DE2-115, indicando os principais componentes.

Fundamentada na imagem publicada em: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502&PartNo=3

1.3. Organização do Trabalho

O trabalho foi organizado da seguinte maneira:

- O Capítulo 2 contém um estudo do neurônio biológico, do neurônio artificial, e de redes neurais *Perceptron*, *Adaline*¸ e redes recorrentes de Hopfield;
- O Capítulo 3 descreve detalhadamente os procedimentos adotados para a síntese da rede *Perceptron* em FPGA e os testes de implementações com diferentes precisões numéricas;
- No Capítulo 4 são expostos e discutidos os resultados obtidos com a síntese da rede *Perceptron* e os testes com variações da precisão numérica;
- No Capítulo 5 encontra-se a conclusão a respeito dos resultados obtidos através dos métodos utilizados;
- O Capítulo 6 propõe futuras atividades que darão continuidade ao presente trabalho.

2. Redes Neurais Artificiais

2.1. O Neurônio Biológico

A teoria de redes neurais baseia-se nos modelos de neurônios biológicos. A primeira descrição detalhada do neurônio biológico é fruto das observações do neurologista espanhol Ramón y Cajal, no século 19. Como ilustrado na Figura 5, embora haja diferentes morfologias de neurônios, todos podem ser divididos em dendritos, soma – ou corpo celular – e axônio (Kovács L. Z., 2006).

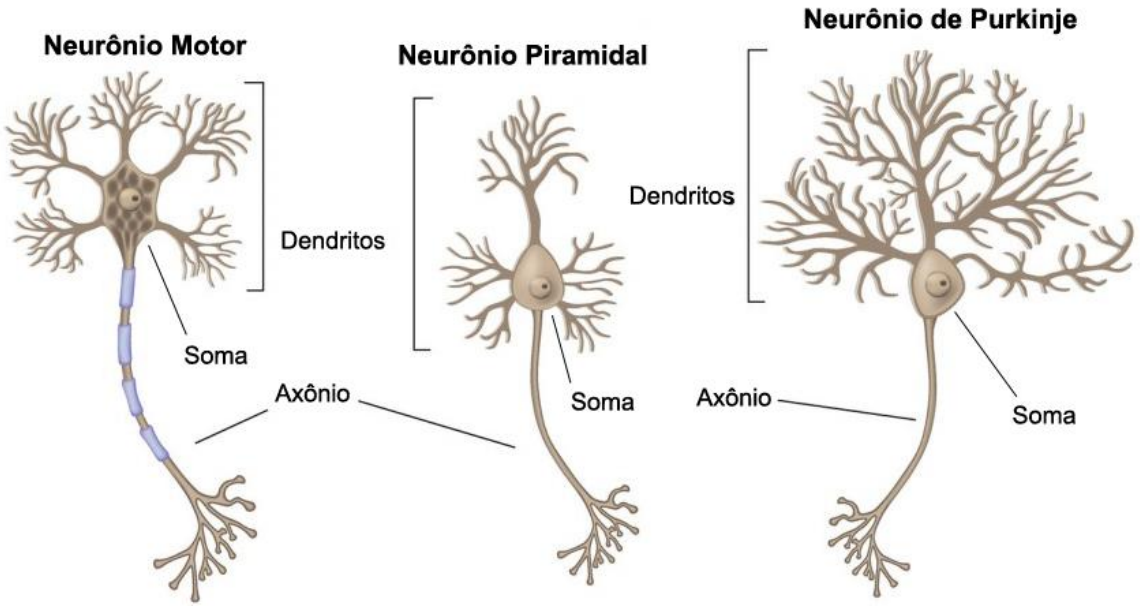


Figura 5 – Exemplos de neurônios biológicos do corpo humano
Fundamentada na imagem publicada em: http://www.interactive-biology.com/3247/the-neuron-external-structure-and-classification/

Comumente, os dendritos apresentam ramificações maiores que o próprio corpo celular, às quais se denominam árvore dendrital, e são responsáveis por receber impulsos de outros neurônios. Por sua vez, o axônio – ou fibra nervosa – é responsável por transmitir os impulsos elétricos a outros neurônios ou ao tecido muscular. A transmissão dos impulsos não é resultado de uma conexão física direta entre os neurônios, mas sim, de conexões sinápticas (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006). As terminações dos axônios possuem uma membrana denominada pré-sináptica, que converte o impulso elétrico proveniente do corpo celular, emitindo neurotransmissores anteriormente armazenados em vesículas. Como ilustrado na Figura 6, os neurotransmissores ligam-se aos receptores da membrana pós-sináptica, presente nos dendritos, estimulando a absorção de íons de sódio e causando um novo pulso elétrico no segundo neurônio, no caso de uma conexão excitatória.

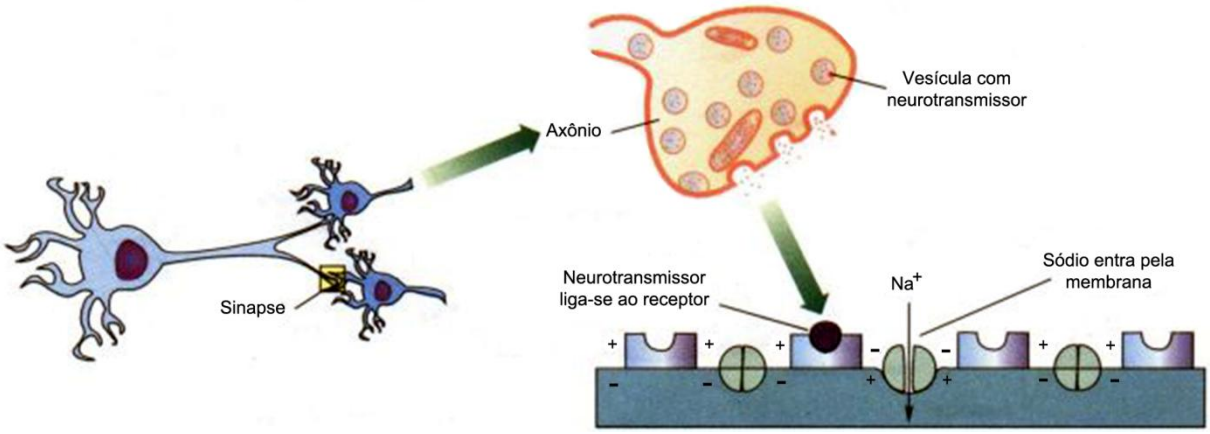


Figura 6 – Conexão sináptica entre neurônios evidenciando a vesícula e a membrana pós-sináptica.

Fundamentada na imagem publicada em: http://www.afh.bio.br/nervoso/nervoso2.asp

Uma conexão excitatória produz uma alteração no potencial elétrico da membrana, que resulta na transmissão de um impulso elétrico. Uma conexão inibitória tem o efeito oposto. Tal comportamento depende dos neurotransmissores envolvidos. A formação de um potencial no axônio do segundo neurônio só ocorre quando os neurotransmissores do primeiro são suficientes para causar uma depolarização na membrana pós-sináptica suficiente para atingir um determinado valor, conhecido como limiar de disparo. O funcionamento de um neurônio está ligado, portanto, à ponderação da intensidade das ligações sinápticas estabelecidas (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006).

A Figura 7 exibe, como exemplo, um modelo de neurônio contendo seis entradas (A,B,C,D,E e F) e uma saída H, sendo a entrada B inibitória. Embora os pulsos de entrada não ocorram simultaneamente, o efeito de cada pulso no neurônio não se degrada instantaneamente e pode durar de milissegundos a segundos, com decaimento aproximadamente exponencial. Este comportamento, conhecido como integração temporal dos estímulos, provoca o resultado exibido na Figura 7 após o recebimento do pulso excitatório em t5, em que o efeito combinado dos pulsos recebidos desde o instante t1 provoca um potencial suficiente para ultrapassar o potencial de ativação e provocar uma saída em H. O efeito combinado dos estímulos é denominado integração espacial dos estímulos, e a combinação destas duas propriedades é denominada integração espacial-temporal dos estímulos (Kovács L. Z., 2006).

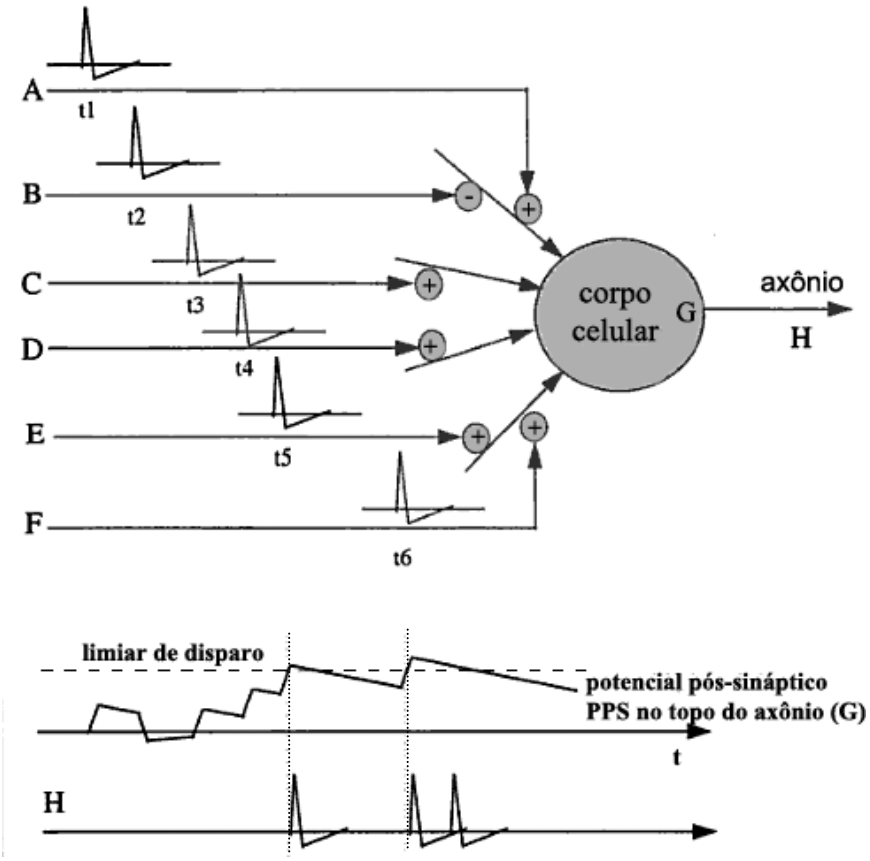


Figura 7 – Análise do funcionamento do modelo de um neurônio com seis entradas (A,B,C,D,E e F) e uma saída H.

Fonte: Kovács L. Z., 1996, página 19

Após o pulso recebido no instante t6, nota-se que foram produzidos dois pulsos na saída H. Este comportamento ocorre devido ao tempo de restabelecimento exigido pelo neurônio. Após a transmissão de um impulso nervoso, o neurônio entra em um estado denominado período de refração absoluta, em que não são emitidos pulsos na saída. Após este período, ocorre o período de refração relativa, em que um pulso é produzido caso haja excitações mais intensas que o limiar de disparo. Portanto, ocorrem dois pulsos após t6, pois a integração espacial-temporal dos pulsos de entrada foi suficiente para manter o potencial pós-sináptico acima do limiar de disparo durante o período de refração absoluta (Kovács L. Z., 2006).

2.2. O Neurônio Artificial

Uma vez que o funcionamento do neurônio biológico foi compreendido, é possível construir um modelo matemático que o represente, como o exibido na Figura 8.

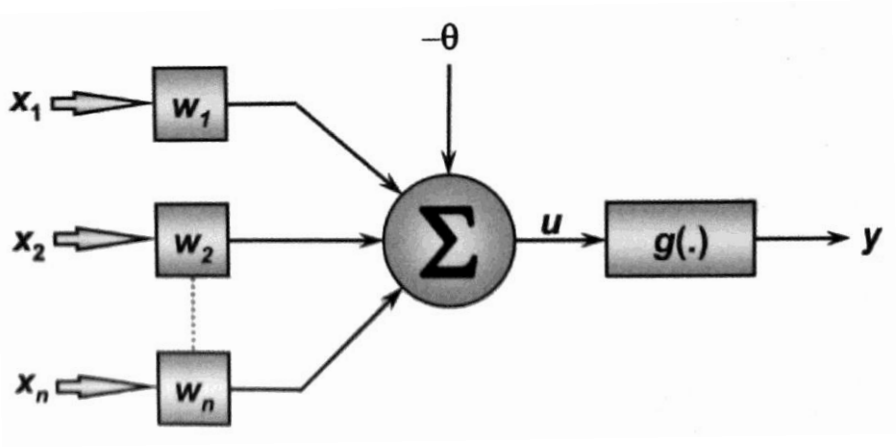


Figura 8 – Modelo matemático de um neurônio.

Fonte: da Silva I. N., Spatti D. H., Flauzino R. A., 2010, página 34

- As entradas $\mathbf{x_1}$ a $\mathbf{x_n}$ representam os pulsos recebidos por neurônios biológicos a partir de outros neurônios ou de fatores externos.
- Os pesos $\mathbf{w_1}$ a $\mathbf{w_n}$ representam a sensibilidade da membrana pós-sináptica dos neurônios biológicos.
- A soma do produto das entradas pelos seus respectivos pesos, efetuada pelo bloco somador Σ, representa a característica de integração espacial do neurônio biológico.
- O parâmetro **θ** representa o limiar de disparo do neurônio biológico, e a subtração deste representa a comparação do potencial pós-sináptico com o limiar de disparo.
- A saída **u** representa o pulso de saída do neurônio biológico, sinal de que o limiar de disparo foi ultrapassado.
- A função **g(.)** representa a concentração de neurotransmissores da membrana pré-sináptica do neurônio biológico.
- A saída **y** equivale aos neurotransmissores ou pulsos elétricos emitidos pelo neurônio biológico.

Este modelo de neurônio foi proposto por McCulloch e Pitts, em 1943. Apesar de ser um modelo simples, engloba todas as características de uma rede neural biológica, e é ainda o modelo mais utilizado em arquiteturas de redes neurais artificiais. O modelo da Figura 8 representa as equações (1) e (2), em que se pode obter a saída **y** do neurônio em função das entradas **x** e dos parâmetros **w** e **θ** através de **u** (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\mathbf{u} = \sum_{i=1}^{n} \mathbf{w_i} \cdot \mathbf{x_i} - \boldsymbol{\theta} \qquad (1)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{u}) \qquad (2)$$

Embora o modelo de um único neurônio seja representado por uma função matemática relativamente simples, a união de vários neurônios, ou seja, a rede neural é capaz de desempenhar funções altamente complexas. A variação dos parâmetros **w** e **θ**, e da função **g(.)** permitem a adequação dos neurônios a problemas distintos. Tal processo, aplicado a toda a RNA é denominado treinamento da rede neural.

Funções de ativação **g(.)** podem ser classificadas quanto à diferenciabilidade como funções parcialmente diferenciáveis – funções que não possuem derivada de primeira ordem – e funções totalmente diferenciáveis – funções que possuem derivada de primeira ordem em todos os pontos em que estão definidas. As principais funções de ativação são classificadas na Tabela 1 (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

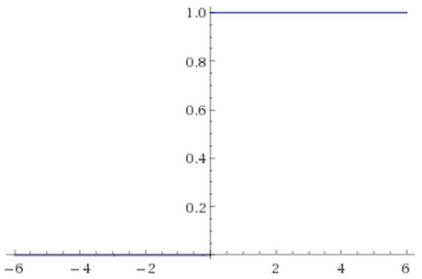
Tabela 1 – Classificação das Funções de Ativação

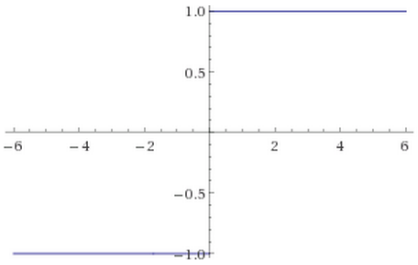
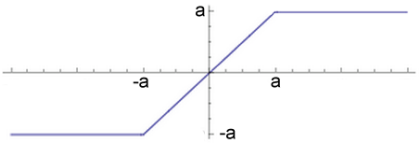
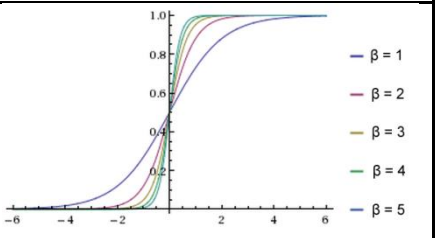
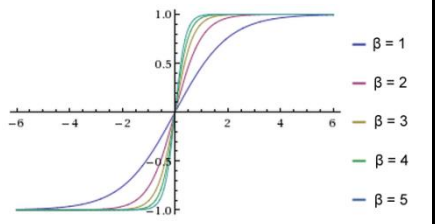
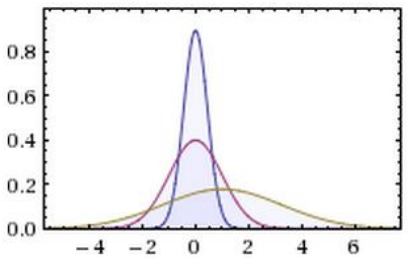
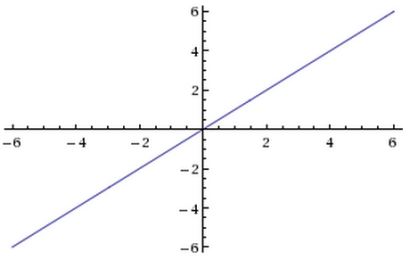
Grupo	Função
Parcialmente Diferenciáveis	Degrau
	Sinal
	Rampa Simétrica
Totalmente Diferenciáveis	Logística
	Tangente Hiperbólica
	Gaussiana
	Linear

As equações que definem cada função com sua respectiva representação gráfica estão reunidas na Tabela 2.

Tabela 2 – Definição das Funções de Ativação

(Gráficos gerados em www.wolframalpha.com)

Função	Notação Matemática	Representação Gráfica
Degrau	$\mathbf{g}(\mathbf{u}) = \begin{cases} 1, & se\ \mathbf{u} \geq 0 \\ 0, & se\ \mathbf{u} < 0 \end{cases}$	

Sinal	$\mathbf{g(u)} = \begin{cases} 1, & se\ \mathbf{u} > 0 \\ 0, & se\ \mathbf{u} = 0 \\ -1, & se\ \mathbf{u} < 0 \end{cases}$ ou $\mathbf{g(u)} = \begin{cases} 1, & se\ \mathbf{u} \geq 0 \\ -1, & se\ \mathbf{u} < 0 \end{cases}$ ou $\mathbf{g(u)} = \begin{cases} 1, & se\ \mathbf{u} > 0 \\ saída\ anterior, & se\ \mathbf{u} = 0 \\ -1, & se\ \mathbf{u} < 0 \end{cases}$	
Rampa Simétrica	$\mathbf{g(u)} = \begin{cases} a, & se\ \mathbf{u} > a \\ \mathbf{u}, & se - a \leq \mathbf{u} \leq a \\ -a, & se\ \mathbf{u} < a \end{cases}$	
Logística	$\mathbf{g(u)} = \frac{1}{1 + e^{-\beta.\mathbf{u}}}$	
Tangente Hiperbólica	$\mathbf{g(u)} = \frac{1 - e^{-\beta.\mathbf{u}}}{1 + e^{-\beta.\mathbf{u}}}$	
Gaussiana	$\mathbf{g(u)} = e^{\frac{-(\mathbf{u}-c)^2}{2\sigma^2}}$	
Linear	$\mathbf{g(u)} = \mathbf{u}$	

2.3. Rede *Perceptron*

A rede *Perceptron* foi idealizada por Rosenblatt no fim da década de 1950. Dando continuidade às idéias de McCulloch, Rosenblatt criou uma rede de múltiplos neurônios ligados a sensores de luz para a identificação de padrões geométricos. A Figura 9 ilustra uma rede neural multicamada, composta por k camadas (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006).

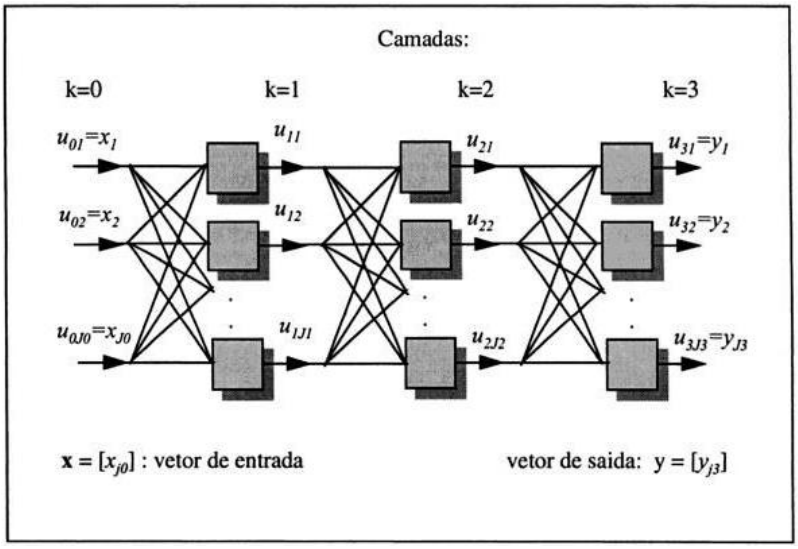


Figura 9 – Rede neural multicamada.

Fonte: Kovács L. Z., 2006, página 40

A camada k=0 é chamada de camada de entrada, a camada k=3 é chamada camada de saída e as camadas intermediárias são chamadas de camadas ocultas (Kovács L. Z., 2006).

A determinação dos parâmetros para a utilização da rede *Perceptron* para a execução das funções booleanas **E** e **OU** de duas variáveis é bastante simples. A proposta de Rosenblatt encontrar um método que permitisse a determinação dos parâmetros da rede de modo que esta pudesse ser utilizada como uma função discriminatória arbitrária. Baseando-se em sistemas biológicos, a forma mais intuitiva de treinamento de uma rede neural consiste no treinamento através de exemplos, em que são apresentadas entradas associadas às suas respectivas saídas, que são repassados até que a rede obtenha o comportamento desejado (Kovács L. Z., 2006).

A rede *Perceptron* de camada simples pode ser descrita matematicamente por **y=g(u)**, onde **g(u)** pode ser a função degrau ou a função sinal, e **u** é definido pela função (3) (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\mathbf{u} = \sum_{i=1}^{n} \mathbf{w_i} \,.\, \mathbf{x_i} - \boldsymbol{\theta} \qquad (3)$$

Portanto, uma rede *Perceptron* de camada simples e duas entradas pode classificar valores separáveis pela reta descrita pela função (4), e ilustrada pela Figura 10 (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\mathbf{w_1}.\mathbf{x_1} + \mathbf{w_2}.\mathbf{x_2} - \boldsymbol{\theta} = 0 \qquad (4)$$

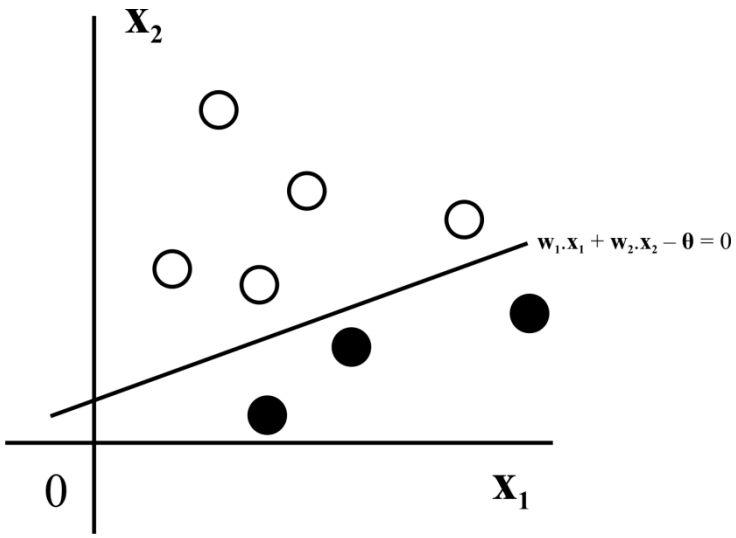


Figura 10 – Representação gráfica do separador linear de uma rede Perceptron de duas entradas.

Fonte: Kovács L. Z., 2006, página 32

A adição de uma entrada à rede resultaria na função (5) que pode ser representada pelo plano da Figura 11 para parâmetros $\mathbf{w_1} = 1$, $\mathbf{w_2} = 1$, $\mathbf{w_3} = -1$ e $\boldsymbol{\theta} = 1$.

$$\mathbf{w_1}.\mathbf{x_1} + \mathbf{w_2}.\mathbf{x_2} + \mathbf{w_3}.\mathbf{x_3} - \boldsymbol{\theta} = 0 \qquad (5)$$

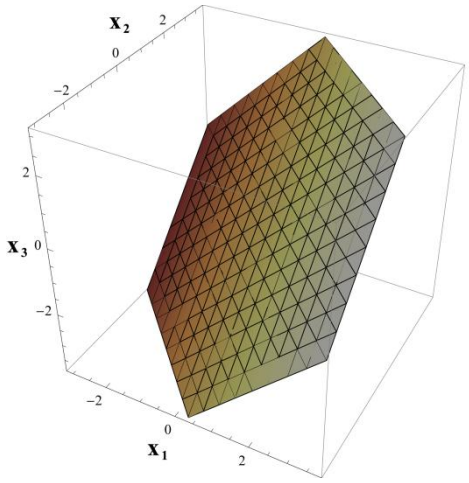


Figura 11 – Representação gráfica do separador linear de uma rede Perceptron de três entradas.

Gerado em www.wolframalpha.com

Observadas tais características da rede *Perceptron* de camada simples, é possível concluir que esta é somente capaz de classificar valores linearmente separáveis, independentemente do número de entradas. O separador linear pode, também, variar de acordo com os parâmetros iniciais adotados para treinamento da rede (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

O treinamento da rede *Perceptron* consiste, portanto, em posicionar o separador linear, até que se obtenha uma fronteira dividindo as duas classes de valores (da Silva I. N., Spatti D. H., Flauzino R. A., 2010). Um algoritmo de treinamento pode ser representado pelas funções (6) e (7) para a determinação de $\mathbf{w_i}$ e $\boldsymbol{\theta}$, respectivamente (Kovács L. Z., 2006).

$$\mathbf{w_i}^{atual} = \mathbf{w_i}^{anterior} + \Delta\mathbf{w_i} \qquad (6)$$

$$\boldsymbol{\theta}^{atual} = \boldsymbol{\theta}^{anterior} - \Delta\boldsymbol{\theta} \qquad (7)$$

Sendo que:

$$\Delta\mathbf{w_i} = \eta.\left(\mathbf{y_l^d} - \mathbf{y_l}\right).\mathbf{x_{i,l}^d} \qquad (8)$$

$$\Delta\boldsymbol{\theta} = \eta.\left(\mathbf{y_l^d} - \mathbf{y_l}\right) \qquad (9)$$

Onde **l** representa a ***l-ésima*** amostra de teste para treinamento da rede, $\mathbf{y}^d$ representa a saída desejada e **y** a saída obtida para uma entrada $\mathbf{x}^d$. O parâmetro η é referido como taxa de aprendizado por estabelecer a taxa com que os ganhos são alterados. Normalmente

compreendido em $0 < \eta < 1$, deve ser adequadamente escolhido para permitir uma rápida convergência dos parâmetros da rede sem que haja instabilidades no processo de treinamento (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006).

Para a resolução de problemas não linearmente separáveis, pode-se utilizar a rede *Perceptron* de várias camadas, no entanto, o processo de treinamento passa a ser mais complexo devido à dificuldade para determinação dos parâmetros das camadas intermediárias – ou ocultas – da rede. Uma opção para o treinamento de uma rede *Perceptron* multicamada é o uso do algoritmo *backpropagation* (Souček B, IRIS Group, 1991).

2.4. Rede *Adaline*

Acrônimo para *(Adaptative Linear Element)*, foi idealizada por Widrow e Hoff em 1960 e tornou-se uma das primeiras aplicações industriais a utilizar redes neurais, no entanto, sua maior contribuição para os estudos de redes neurais foi a introdução do algoritmo de aprendizado denominado regra *Delta*, por ser o precursor da regra *Delta* generalizada, utilizada para o treinamento de redes *Perceptron* multicamadas (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006).

A rede *Adaline* consiste em uma rede de apenas uma camada de neurônios. A composição de várias *Adaline* formando uma rede multicamada é denominada *Madaline* (Múltipla *Adaline*) (da Silva I. N., Spatti D. H., Flauzino R. A., 2010)(Kovács L. Z., 2006).

Ilustrada pela Figura 12, a rede, analogamente à rede *Perceptron*, pode ser descrita matematicamente por **y=g(u)**, onde **g(u)** pode ser a função degrau ou a função sinal, e **u** é definido pela função (10) (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\mathbf{u} = \sum_{i=1}^{n} \mathbf{w_i} \,.\, \mathbf{x_i} - \boldsymbol{\theta} \qquad \mathbf{(10)}$$

Ou, sendo $\boldsymbol{\theta} = \mathbf{w_0}$:

$$\mathbf{u} = \sum_{i=0}^{n} \mathbf{w_i} \,.\, \mathbf{x_i} \qquad \mathbf{(11)}$$

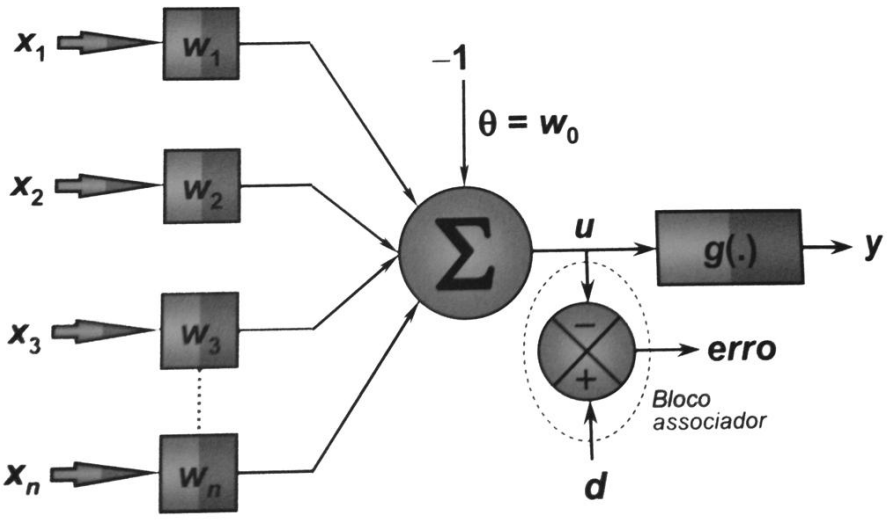


Figura 12 – Rede Adaline.

Fonte: da Silva I. N., Spatti D. H., Flauzino R. A., 2010, página 74

O bloco associador, observado na Figura 12, tem a função de auxiliar no processo de ajuste dos pesos $\mathbf{w_i}$ durante o treinamento da rede (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

A principal diferença entre as redes *Adaline* e *Perceptron* é o processo de treinamento. A rede *Adaline* utiliza um algoritmo baseado na minimização do erro quadrático entre a saída desejada **d** e a saída **u** obtida. A atualização dos parâmetros $\mathbf{w_i}$ ocorre como descrito pelas equações (12) e (13), para **p** amostras de testes para treinamento da rede (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\mathbf{w_i}^{atual} = \mathbf{w_i}^{anterior} + \Delta\mathbf{w_i} \qquad (12)$$

$$\Delta\mathbf{w_i} = \eta . \sum_{k=1}^{\mathbf{p}} \left(\mathbf{d}^{(k)} - \mathbf{u}\right) . \mathbf{x_i}^{(k)} \qquad (13)$$

Por basear-se na redução do erro pelo método dos mínimos quadrados, o separador linear da rede *Adaline* será sempre o mesmo, independente dos parâmetros iniciais utilizados para o treinamento da rede (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

2.5. Redes Recorrentes de Hopfield

Em redes recorrentes, as saídas de uma camada podem ser realimentadas às suas entradas. As principais redes recorrentes são as denominadas redes de Hopfield, em referência ao seu idealizador Hopfield, cujos trabalhos realizados em 1982 contribuíram para renovar os interesses por pesquisas em redes neurais, estagnadas desde a publicação do livro *Perceptron* por Minsky e Papert em 1969 (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

A Figura 13 exibe a proposta original para a rede de Hopfield, onde $\mathbf{z^{-1}}$ indica atraso temporal de uma unidade e i_n^b é o limiar de disparo do *n-ésimo* neurônio. Observa-se que a rede é composta de uma única camada, e todas as saídas realimentam todas as entradas.

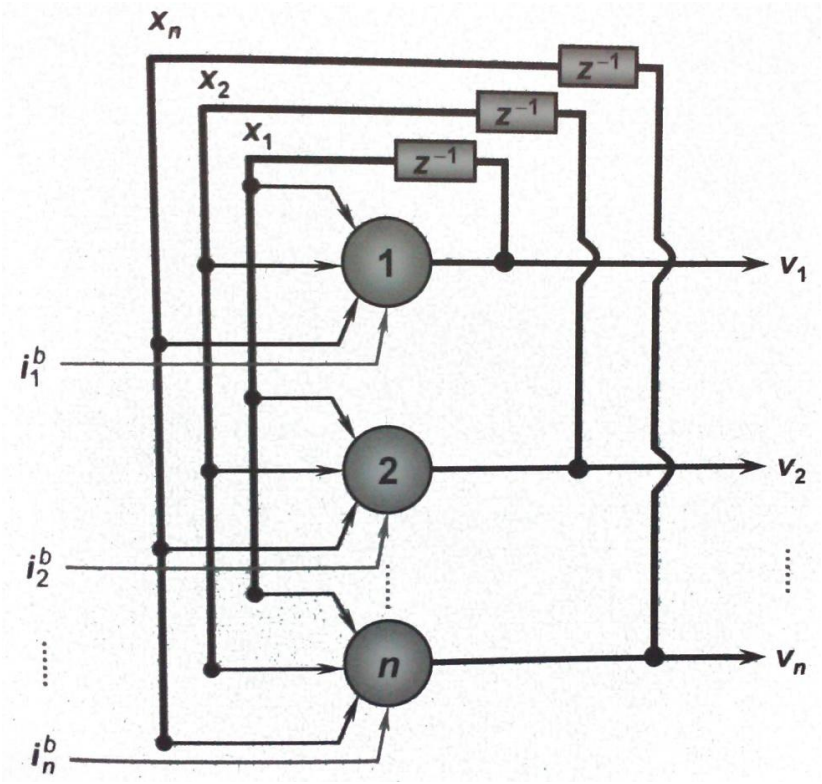


Figura 13 – Rede de Hopfield.

Fonte: da Silva I. N., Spatti D. H., Flauzino R. A., 2010, página 201

O funcionamento da rede pode ser representado pelas funções (14) e (15) (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

$$\dot{\mathbf{u}}_j(t) = -\eta.\mathbf{u}_j(t) + \sum_{i=1}^{n} \mathbf{w}_{ji}.\mathbf{v}_i(t) + \mathbf{i}_j^b, onde\ j = 1, \dots, n \qquad \textbf{(14)}$$

$$\mathbf{v}_j(t) = \mathbf{g}(\mathbf{u}_j(t)) \qquad \textbf{(15)}$$

Onde:

- $\mathbf{u}_j(t)$ é o estado interno do *i-ésimo* neurônio, com $\mathbf{u}_j(t) = d\mathbf{u}/dt$;
- $\mathbf{v}_j(t)$ é a saída do *j-ésimo* neurônio;
- $\mathbf{w}_{ji}$ é o valor do peso sináptico conectando o *j-ésimo* neurônio ao *i-ésimo* neurônio;
- i_j^b é o limiar (bias) aplicado ao *j-ésimo* neurônio;
- $\mathbf{g}(.)$ é uma função monótona crescente, que limita a saída de cada neurônio a um intervalo pré-definido;
- $\eta.\mathbf{u}_j(t)$ é um termo de decaimento passivo;

Portanto, a entrada inicial **x**(t_0) gera a saída inicial **v**(t_0). Nos próximos passos, a rede retorna a saída à entrada até que a saída da rede convirja para um valor.

Por se tratarem de redes dinâmicas, as redes de Hopfield nem sempre convergem. Para se obter a convergência, devem-se escolher adequadamente os parâmetros, de modo que suas derivadas temporais sejam iguais ou menores que zero. Para tanto, pode-se utilizar a função de Lyapunov.

3. Metodologia

3.1. *Kit* DE2-115

O *kit* de desenvolvimento DE2-115 possui, como principal componente, um FPGA Cyclone IV, EP4CE115F29C7N, que possui 114.480 elementos lógicos, 3.888 *Kbits* de memória interna, 266 multiplicadores de 18 x 18 *bits* 4 *phase-locked* loops de uso genérico e 528 entradas/saídas disponíveis para o usuário (Terasic Technologies Inc.). Tais características estão muito além do necessário para esta aplicação, no entanto, a intercambiabilidade oferecida pela VHDL permite que o circuito seja sintetizado em centenas de dispositivos diferentes.

Devido à grande variedade de interfaces, o *kit* DE2-115 pode ser utilizado para o desenvolvimento de diversas aplicações, completamente distintas, como processamento áudio e vídeo e aplicações com interface *ethernet*.

Como exibido na Figura 4, o *kit* possui 18 chaves deslizantes (**S0** a **S17**) e um *LED* vermelho (***LED* R0** a ***LED* R17**) acima de cada uma delas, dos quais serão utilizadas 16 chaves deslizantes e 16 *LEDs*, conforme a Tabela 3, onde $\mathbf{x_i}(0)$ representa o *bit* menos significativo da entrada $\mathbf{x_i}$.

Tabela 3 – Relação das Chaves com as Respectivas Entradas

***LED* R17**	***LED* R16**	***LED* R15**	***LED* R14**	***LED* R13**	***LED* R12**	***LED* R11**	***LED* R10**	***LED* R9**	***LED* R8**	***LED* R7**	***LED* R6**	***LED* R5**	***LED* R4**	***LED* R3**	***LED* R2**	***LED* R1**	***LED* R0**
S17	**S16**	**S15**	**S14**	**S13**	**S12**	**S11**	**S10**	**S9**	**S8**	**S7**	**S6**	**S5**	**S4**	**S3**	**S2**	**S1**	**S0**
$\mathbf{x_1}(7)$	$\mathbf{x_1}(6)$	$\mathbf{x_1}(5)$	$\mathbf{x_1}(4)$	$\mathbf{x_1}(3)$	$\mathbf{x_1}(2)$	$\mathbf{x_1}(1)$	$\mathbf{x_1}(0)$	NC	NC	$\mathbf{x_2}(7)$	$\mathbf{x_2}(6)$	$\mathbf{x_2}(5)$	$\mathbf{x_2}(4)$	$\mathbf{x_2}(3)$	$\mathbf{x_2}(2)$	$\mathbf{x_2}(1)$	$\mathbf{x_2}(0)$

O ***LED G8*** está diretamente conectado à saída **y**, que também é representada pelos *displays* de sete segmentos *HEX0*, *HEX1* e *HEX2*, de forma descrita na seção 3.4 (pág. 28).

A entrada *Blaster* do *kit* é conectada ao computador por um cabo USB para a configuração do FPGA. O *software* para projeto e síntese, Quartus II versão 10.0 é fornecido com o *kit* DE2-115 e possui versão gratuita sem prazo de expiração.

A relação completa das portas existentes no circuito da rede com os respectivos pinos do FPGA encontra-se na Tabela 13 (seção 8.2, pág. 47).

3.2. Rede *Perceptron* Utilizada

Com objetivo de obter uma rede neural já treinada e com valores de entrada e saída confiáveis para averiguação do seu correto funcionamento, utilizou-se a rede neural implementada por Davi Laraia Colman, descrita em seu Relatório Final de Iniciação Científica: Projeto de Redes Neurais Artificiais Aplicadas em Problemas de Automação Industrial (Colman D. L., 2006). Neste trabalho realizado no ano de 2006, sob a orientação do Professor Doutor Ivan Nunes da Silva, o autor sintetizou uma rede neural do tipo *Perceptron* de camada simples e

duas entradas, destinada ao controle direto de torque de um motor de indução, em um FPGA utilizando descrição do circuito por diagrama de blocos.

A Tabela 4 contém os parâmetros da rede e a Tabela 5 contém as entradas para teste de funcionamento da rede com suas saídas correspondentes (Colman D. L., 2006).

Tabela 4 - Parâmetros da Rede *Perceptron*

Parâmetro	Valor
Peso sináptico w_1	3,23
Peso sináptico w_2	-5,42
Limiar de disparo θ	0,83

Tabela 5 – Dados de Teste e Resultados

Entrada x_1	Entrada x_2	Saída y Teórica
0,93	0,78	0
1,00	0,68	0
0,50	0,07	1
0,29	0,01	1
0,67	0,23	1
0,99	0,28	1
0,02	0,77	0
0,82	0,31	1
0,26	0,83	0
0,60	0,91	0

A Figura 14 ilustra a representação gráfica da rede *Perceptron* utilizada.

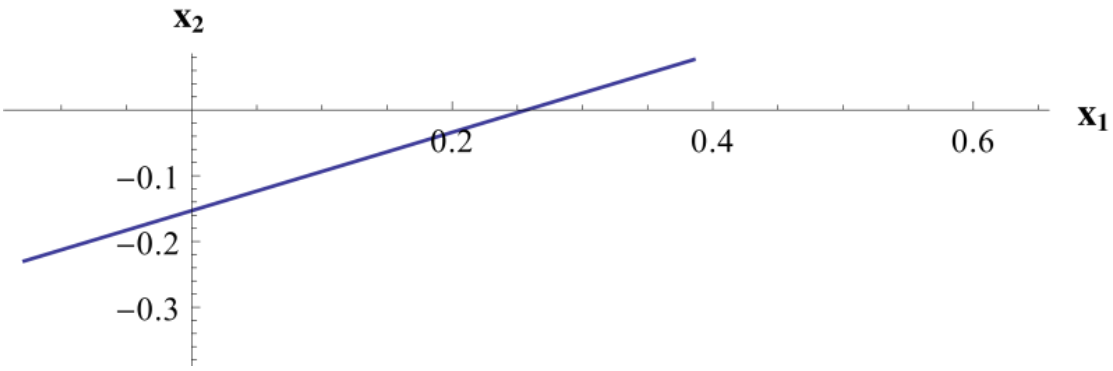


Figura 14 – Representação gráfica do separador linear da rede Perceptron utilizada. Gerado em www.wolframalpha.com

3.3. Adequação da Rede

As entradas e parâmetros foram representados no circuito por notação binária de ponto fixo. Foram sintetizadas redes com diferentes precisões na representação numérica a fim de comparar a influência destas no número de elementos lógicos exigidos do FPGA. A representação para a qual a rede foi configurada no FPGA e testada para todas as entradas da Tabela 5 encontra-se na Tabela 6. O separador decimal presente na representação binária tem o objetivo de facilitar a inter-relação dos dados da tabela, e não é declarado na implementação prática do circuito.

Tabela 6 – Representação Binária dos Dados de Teste

x_1 (Decimal)	x_1 (Binária)	x_2 (Decimal)	x_2 (Binária)
0,93	0,1110111	0,78	0,1100011
1,00	1,0000000	0,68	0,1010111
0,50	0,1000000	0,07	0,0001000
0,29	0,0100101	0,01	0,0000001
0,67	0,1010101	0,23	0,0011101
0,99	0,1111110	0,28	0,0100011
0,02	0,0000010	0,77	0,1100010
0,82	0,1101000	0,31	0,0100111
0,26	0,0100001	0,83	0,1101010
0,60	0,1001100	0,91	0,1110100

Nesta representação, os valores decimais $\mathbf{d_b}$ são relacionados aos valores binários **B** em função de cada um dos *bits* $\mathbf{b_i}$ de acordo com a equação (16), onde $\mathbf{b_7}$ é o *bit* mais significativo de **B**.

$$\mathbf{d_b} = \sum_{i=0}^{7} \mathbf{b_i} \cdot 2^{i-7} \qquad \mathbf{(16)}$$

A escolha de um único *bit* para a parte inteira e sete para a parte fracionária resultou da observação da amplitude dos valores de entrada. Como a parte inteira dos valores de entrada varia entre "0" e "1", um *bit* é suficiente para representá-lo. Quanto à parte fracionária, pode-se verificar o erro relativo de aproximação $\mathbf{e_{\%}}$, causado pela representação utilizada, pela equação (17), em que **d** é o valor original da entrada, e $\mathbf{d_b}$ é o valor decimal convertido a partir do valor binário, de acordo com (16).

$$\mathbf{e}_{\%} = \frac{(\mathbf{d} - \mathbf{d_b})}{\mathbf{d}} . 100 \qquad (17)$$

A comparação dos resultados é exibida na Tabela 7, onde nota-se que, exceto em três casos, o erro permaneceu abaixo de 2,5%. Observa-se também, que o erro foi maior nos casos em que a entrada apresentava valores próximos ao extremo inferior da amplitude total de valores. No entanto, estes casos refletem também em um peso pouco expressivo no funcionamento da rede neural. Um erro maior de aproximação é, portanto, tolerável para tais valores, e a representação numérica é suficiente para o problema.

Tabela 7 – Erro Devido à Aproximação na Representação Numérica

x_1 (d)	x_1 (d_b)	$e_{\%}$ de x_1	x_2 (d)	x_2 (d_b)	$e_{\%}$ de x_2
0,93	0,9296	0,04	0,78	0,7734	0,85
1,00	1,0000	0,00	0,68	0,6796	0,06
0,50	0,5000	0,00	0,07	0,0625	10,71
0,29	0,2890	0,34	0,01	0,0078	22,00
0,67	0,6640	0,90	0,23	0,2265	1,52
0,99	0,9843	0,58	0,28	0,2734	2,36
0,02	0,0156	22,00	0,77	0,7656	0,57
0,82	0,8125	0,91	0,31	0,3047	1,71
0,26	0,2578	0,85	0,83	0,8281	0,23
0,60	0,5937	1,05	0,91	0,9062	0,42

O parâmetro $\boldsymbol{\theta}$ possui valor 0,83 e, portanto, o erro para a mesma representação utilizada para as entradas é de 0,23%. Já os parâmetros $\mathbf{w_1}$ e $\mathbf{w_2}$ apresentam valores maiores que qualquer um dos valores apresentados pelas entradas, exigindo mais *bits* para a representação da parte inteira. A Tabela 8 exibe a representação binária dos parâmetros $\mathbf{w_1}$ e $\mathbf{w_2}$ associada ao respectivo erro de aproximação.

Tabela 8 – Representação Binária dos Parâmetros e Erros de aproximação

Parâmetro	Valor Binário	$e_{\%}$
Peso sináptico w_1	011,0011101	0,11
Peso sináptico w_2	101,0110101	0,11

Observa-se que, embora a representação dos parâmetros $\mathbf{w_1}$ e $\mathbf{w_2}$ apresente dois *bits* a mais que a representação das entradas $\mathbf{x_1}$ e $\mathbf{x_2}$, o número de *bits* utilizados para a representação da parta fracionária continua a mesma e, portanto, não há necessidade de modificar a representação das entradas, pois a adição de dois *bits* "0" à esquerda não alteraria o valor por elas representado. No entanto, a existência de mais dois *bits* na representação dos parâmetros resultaria em um maior consumo de recursos do FPGA, uma vez que o bloco multiplicador deveria executar a operação nestes *bits*, pois ao se representar o número de *bits* dos pesos $\mathbf{w_1}$ e $\mathbf{w_2}$ por $\mathbf{N_w}$ e o número de *bits* das entradas $\mathbf{x_1}$ e $\mathbf{x_2}$ por $\mathbf{N_x}$, o número binário resultante do produto $\mathbf{w_i} \cdot \mathbf{x_i}$ será formado por $\mathbf{N_w} + \mathbf{N_x}$ *bits*. Portanto, no caso adotado, em que $\mathbf{x_1}$ e $\mathbf{x_2}$ são compostos por oito *bits*, o produto $\mathbf{p_i}$ formado por $\mathbf{w_i} \cdot \mathbf{x_i}$ é composto por 18 *bits*. Se $\mathbf{x_1}$ e $\mathbf{x_2}$ fossem compostos por dez *bits*, seriam necessários 20 *bits* para representar $\mathbf{p_i}$.

Pode-se observar que o sinal negativo do parâmetro $\mathbf{w_2}$ foi omitido. Para esta rede, não é necessária uma representação de dados com *bit* de sinal, pois somente o parâmetro é negativo e o sinal pode ser transferido para a operação de adição, no momento da soma, ação que não seria possível caso as variáveis de entrada $\mathbf{x_1}$ e $\mathbf{x_2}$ alternassem entre valores positivos e negativos. Neste caso, seria necessário adotar uma representação binária dotada de um *bit* de sinal. A representação binária de ponto fixo com complemento de dois, associada ao algoritmo de Booth's para a operação de multiplicação é uma opção eficiente para a implementação de redes neurais (Hariprasath S., Prabakar T. N., 2012).

Neste caso, há ainda mais uma observação a ser feita com o objetivo de otimizar a rede para a síntese em hardware. Note que a saída $\mathbf{y}$ é descrita conforme a equação (18):

$$\mathbf{y} = \begin{cases} 1\,, & se \sum_{i=1}^{n} \mathbf{w_i} \cdot \mathbf{x_i} - \boldsymbol{\theta} \geq 0 \\ 0\,, & se \sum_{i=1}^{n} \mathbf{w_i} \cdot \mathbf{x_i} - \boldsymbol{\theta} < 0 \end{cases} \qquad \textbf{(18)}$$

E, portanto, para este caso deve-se calcular:

$$\mathbf{y} = \begin{cases} 1, & se\ \mathbf{w_1} \cdot \mathbf{x_1} - \mathbf{w_2} \cdot \mathbf{x_2} - \boldsymbol{\theta} \geq 0 \\ 0, & se\ \mathbf{w_1} \cdot \mathbf{x_1} - \mathbf{w_2} \cdot \mathbf{x_2} - \boldsymbol{\theta} < 0 \end{cases} \qquad \textbf{(19)}$$

ou

$$\mathbf{y} = \begin{cases} 1, & se\ \mathbf{w_1} \cdot \mathbf{x_1} - \boldsymbol{\theta} \geq \mathbf{w_2} \cdot \mathbf{x_2} \\ 0, & se\ \mathbf{w_1} \cdot \mathbf{x_1} - \boldsymbol{\theta} < \mathbf{w_2} \cdot \mathbf{x_2} \end{cases} \qquad \textbf{(20)}$$

Note que a equação (20) possui uma operação de subtração, a menos que a equação (19), resultando em maior simplicidade do circuito gerado, e menor número de elementos lógicos para sintetizá-lo em um FPGA. De fato, a observação das equações que descrevem os circuitos, e a faixa de valores das variáveis é uma forma importante de otimizar redes neurais, principalmente em aplicações com centenas de entradas, e várias camadas de neurônios, em que o consumo de recursos passa a ser crítico. Devido à grande relevância do aumento no número de *bits* em grandes redes neurais, utilizam-se entradas com valores normalizados (da Silva I. N., Spatti D. H., Flauzino R. A., 2010).

3.4. Descrição da Rede em VHDL

O circuito da RNA foi descrito e sintetizado utilizando-se o *software* Quartus II, versão 10.0 da Altera através de VHDL.

A entidade descrita para o projeto pode ser observada na Figura 15, em que foram evidenciados os parâmetros $\mathbf{w_1}$ e $\mathbf{w_2}$, o parâmetro **t,** representando **θ,** as entradas $\mathbf{x_1}$ e $\mathbf{x_2}$ e a saída **y**.

O tipo genérico **bx** é declarado como o número de *bits* que formam as estradas $\mathbf{x_1}$ e $\mathbf{x_2}$. O tipo **bxw** é declarado como o número de *bits* que formam os sinais **xw1**, **xw2** e **soma**, exibidos na Figura 16.

Os demais tipos genéricos e as portas “disp1”, “disp2”, “disp3” e “disp4” são utilizados para representar a saída da rede nos *displays* de sete segmentes, da forma **P = Y**, onde **Y** exibe valor “0” ou “1”, de acordo com o valor da saída **y** da rede.

As portas “LEDx1” e “LEDx2” exibem o valor dos *bits* que compõem, respectivamente, a entrada $\mathbf{x_1}$ (LEDR10 a LEDR17) e a entrada $\mathbf{x_2}$ (LEDR0 a LEDR7).

```
 7  ENTITY rede IS
 8
 9     GENERIC(bx     : NATURAL                      := 7;
10             bxw    : NATURAL                      := 17;
11             w1     : STD_LOGIC_VECTOR(9 DOWNTO 0) := "0110011101";
12             w2     : STD_LOGIC_VECTOR(9 DOWNTO 0) := "1010110101";
13             t      : STD_LOGIC_VECTOR(7 DOWNTO 0) := "01101010";
14             apaga  : STD_LOGIC_VECTOR(0 TO 6)     := "1111111";
15             P      : STD_LOGIC_VECTOR(0 TO 6)     := "0011000";
16             igual  : STD_LOGIC_VECTOR(0 TO 6)     := "1110110"
17
18     );
19
20     PORT    (x1,x2                    : IN     STD_LOGIC_VECTOR(bx DOWNTO 0);
21              LEDx1,LEDx2              : OUT    STD_LOGIC_VECTOR(bx DOWNTO 0);
22              disp1,disp2,disp3,disp4  : OUT    STD_LOGIC_VECTOR(0 TO 6);
23              y                        : BUFFER STD_LOGIC
24     );
25
26  END rede;
```

Figura 15 – Entidade da Rede *Perceptron*.

É importante observar a praticidade resultante da criação de tipos genéricos, que permite o rápido acesso a valores de constantes no arquivo de descrição de *hardware*, e sua fácil alteração. A Figura 16 exibe a arquitetura descrita para a rede neural.

A utilização das constantes **bx** e **bxw** tem o objetivo de facilitar a declaração dos respectivos sinais e portas relacionados durante os testes apresentados na seção 3.5 (pág. 30).

```
28  ARCHITECTURE arch OF rede IS
29
30  COMPONENT decoder IS
31
32      PORT  (INT        : IN  NATURAL RANGE 0 TO 13;
33             DISPLAY      : OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
34
35  END COMPONENT;
36
37  SIGNAL xw1, xw2, soma          : STD_LOGIC_VECTOR((bxw) DOWNTO 0);
38  SIGNAL dig                     : NATURAL RANGE 0 TO 13;
39
40  BEGIN
41
42     decode   : decoder   PORT MAP (dig,disp1);
43
44     dig <= conv_integer(y);
45     disp4 <= apaga;
46     disp3 <= P;
47     disp2 <= igual;
48
49     LEDx1 <= x1;
50     LEDx2 <= x2;
51
52     xw1 <= w1*x1;
53     xw2 <= w2*x2;
54
55     soma <= xw1 - t;
56
57     y <=  '0' when soma <  xw2 else
58           '1' when soma >=  xw2;
59
60  END arch;
```

Figura 16 – Arquitetura da Rede *Perceptron*.

O componente ***decoder***, cuja descrição completa pode ser encontrada na seção 8.1 (pág. 47), é um decodificador de números inteiros para um *display* de sete segmentos, e é utilizado para exibir o valor da saída **y**. A possibilidade de utilizar projetos inteiros como componentes de outros projetos pode ser de grande vantagem no projeto de redes neurais com várias camadas, em que cada camada pode ser criada em um projeto individual, e posteriormente utilizada repetidamente como componente para gerar uma rede complexa.

Também é importante notar que, embora seja necessário especificar as dimensões de portas, sinais e constantes, o operador de produto é indiferente à quantidade de *bits* dos operandos. Esta característica representa uma simplificação em relação à descrição de circuitos com diagramas de blocos, em que o número de *bits* das entradas e saídas dos blocos multiplicadores precisam ser especificados e alterados a cada mudança na dimensão dos operadores.

As bibliotecas utilizadas e suas respectivas funções na síntese do circuito são listadas na Tabela 9 (The Institute of Electrical and Electronics Engineers, Inc, 1997).

Tabela 9 – Bibliotecas Utilizadas.

Nome da Biblioteca	Função
IEEE Std. 1164	Contém as demais bibliotecas.
IEEE Std. Logic Unsigned	Contém o padrão numérico para a representação de um *bit (std_logic)* e de um vetor de *bits (std_logic_vector)*. Não há representação de valores negativos.
IEEE Std. Logic Arith	Define operações aritméticas básicas (operadores "+","-","*","/").

As adequações apresentadas na seção 3.3 (pág. 25) permitiram o uso da biblioteca *IEEE Std. Logic Unsigned*, pois eliminaram a necessidade de representação de números negativos. Caso fosse necessária tal representação, poderia ser utilizada a biblioteca *IEEE Std. Logic Signed*, que utiliza a representação por complemento de dois para operações com números negativos (The Institute of Electrical and Electronics Engineers, Inc, 1997).

3.5. Variações no Número de *Bits*

Embora a representação numérica empregada tenha se mostrado satisfatória, conforme as justificativas apresentadas na seção 3.3 (pág 25), foram sintetizadas variações do circuito apresentado, alterando-se somente o número de *bits* das entradas e parâmetros. Tais variações permitiram observar a relação entre um número maior de *bits* na representação numérica e o número de elementos lógicos exigidos na síntese do circuito da rede neural em FPGA.

A Tabela 10 exibe o número de *bits* das entradas ($\mathbf{x_1}$ e $\mathbf{x_2}$) e o número de *bits* e valor das constantes ($\mathbf{w_1}$, $\mathbf{w_2}$ e $\mathbf{t}$), que tiveram seu valor binário completado somente com *bits* "1" ou somente com *bits* "0" até que se obtivessem o número de *bits* desejados no teste. O teste 1.0 é o caso inicial, e no teste 1.1 foi sintetizado um circuito com menor precisão na representação numérica.

Tabela 10 – Variação do Número de *Bits* para Representação Numérica.

Número do Teste	Entrada / Parâmetro	Valor Binário (constantes)	Número de *bits*
1.0	x_1	-	8
	x_2	-	8
	t	0,1101010	8
	w_1	011,0011101	10
	w_2	101,0110101	10
1.1	x_1	-	6
	x_2	-	6
	t	0,11010	6
	w_1	011,00111	8
	w_2	101,01101	8
1.2	x_1	-	10
	x_2	-	10
	t	0,110101011	10
	w_1	011,001110111	12
	w_2	101,011010111	12
1.3	x_1	-	12
	x_2	-	12
	t	0,11010101111	12
	w_1	011,00111011111	14
	w_2	101,01101011111	14
1.4	x_1	-	14
	x_2	-	14
	t	0,110101011111	14
	w_1	011,001110111111	16
	w_2	101,011010111111	16
1.5	x_1	-	10
	x_2	-	10
	t	0,110101000	10
	w_1	011,001110100	12
	w_2	101,011010100	12
1.6	x_1	-	12
	x_2	-	12
	t	0,11010100000	12
	w_1	011,00111010000	14
	w_2	101,01101010000	14
1.7	x_1	-	14
	x_2	-	14
	t	0,1101010000000	14
	w_1	011,0011101000000	16
	w_2	101,0110101000000	16

4. Resultados

4.1. Síntese

O circuito descrito pela Figura 15 e pela Figura 16, na seção 3.4 (pág. 28), foi sintetizado no FPGA Cyclone IV, EP4CE115F29C7N, através do software Quartus II 10.0. O circuito equivalente à descrição de hardware, gerado pelo próprio Quartus II é exibido na Figura 17.

A Tabela 11 compara os dados apresentados na Tabela 5 com a saída obtida no circuito sintetizado.

Tabela 11 – Verificação da Rede *Perceptron* Sintetizada.

Número do Teste	Entrada x_1	Entrada x_2	Saída y Teórica	Saída y Obtida
1	0,93	0,78	0	0
2	1,00	0,68	0	0
3	0,50	0,07	1	1
4	0,29	0,01	1	1
5	0,67	0,23	1	1
6	0,99	0,28	1	1
7	0,02	0,77	0	0
8	0,82	0,31	1	1
9	0,26	0,83	0	0
10	0,60	0,91	0	0

A Figura 18 exibe o kit DE2-115 da Terasic com o circuito sintetizado, durante a etapa de teste número 1 da Tabela 11. É possível observar os valores inseridos para as entradas x1 (01110111) e x2 (01100011) através da posição das chaves e do estado dos LEDs vermelhos, e a saída y = 0 exibido pelo LED verde e pelos displays de sete segmentos (P=0).

A Figura 19 exibe o circuito sintetizado durante a etapa de teste número 8 da Tabela 11. Neste caso, os valores inseridos através das chaves e exibidos nos *LEDs* vermelhos para a entrada $\mathbf{x_1}$ é 01101000 e para $\mathbf{x_2}$ é 00100111. O *LED* verde aceso e o valor exibido pelos *displays* de sete segmentos (P=1) indicam a saída **y** = 1.

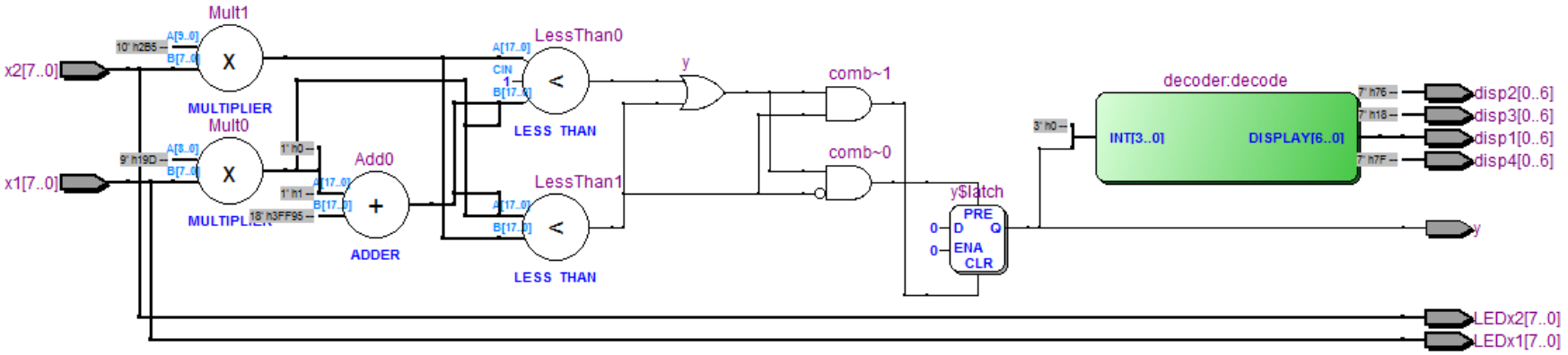


Figura 17 – Circuito equivalente à descrição da rede *Perceptron*.

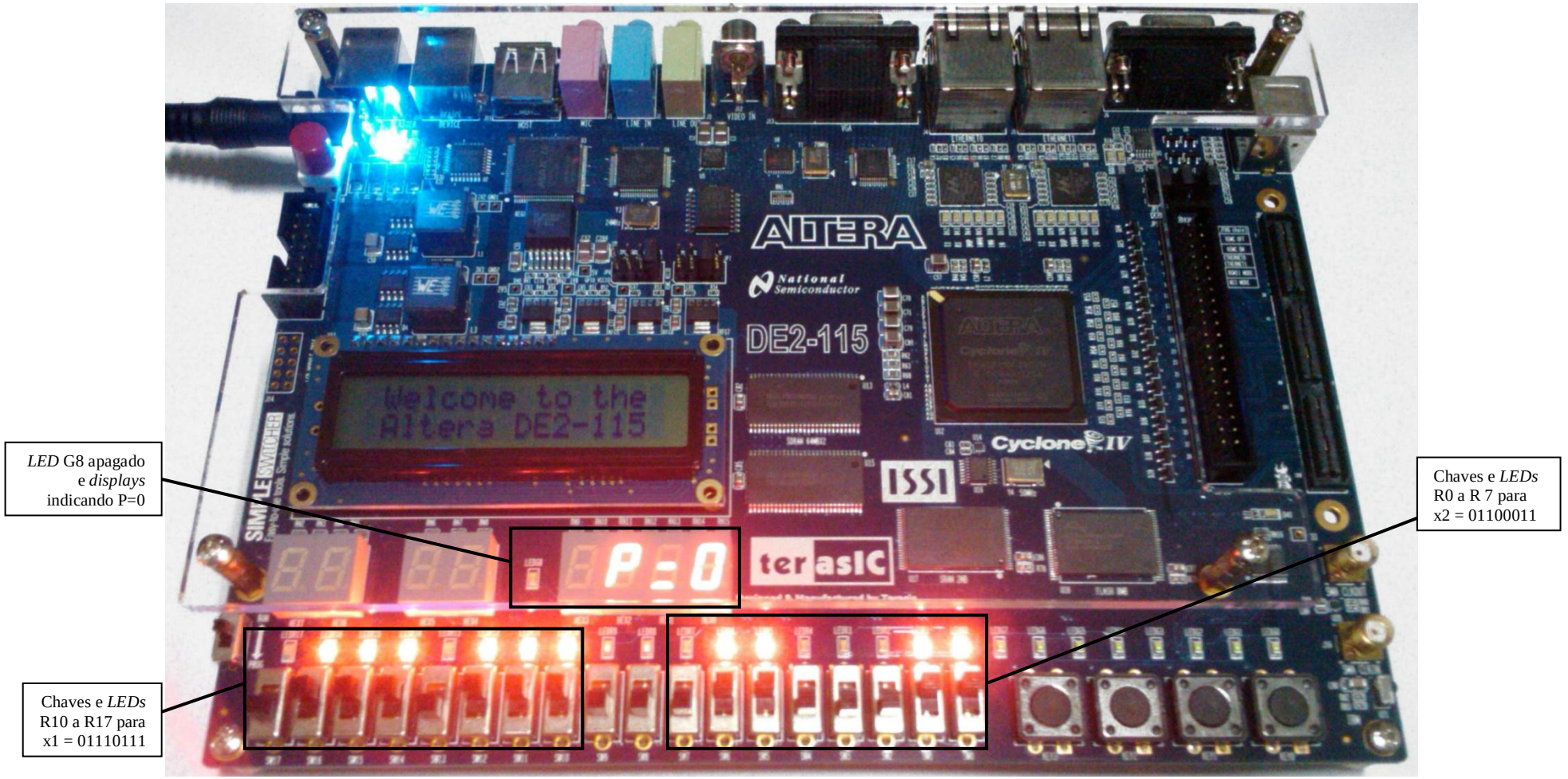


Figura 18 – Circuito da rede *Perceptron* implementado no *kit* DE2-115 (Teste 1 da Tabela 11).

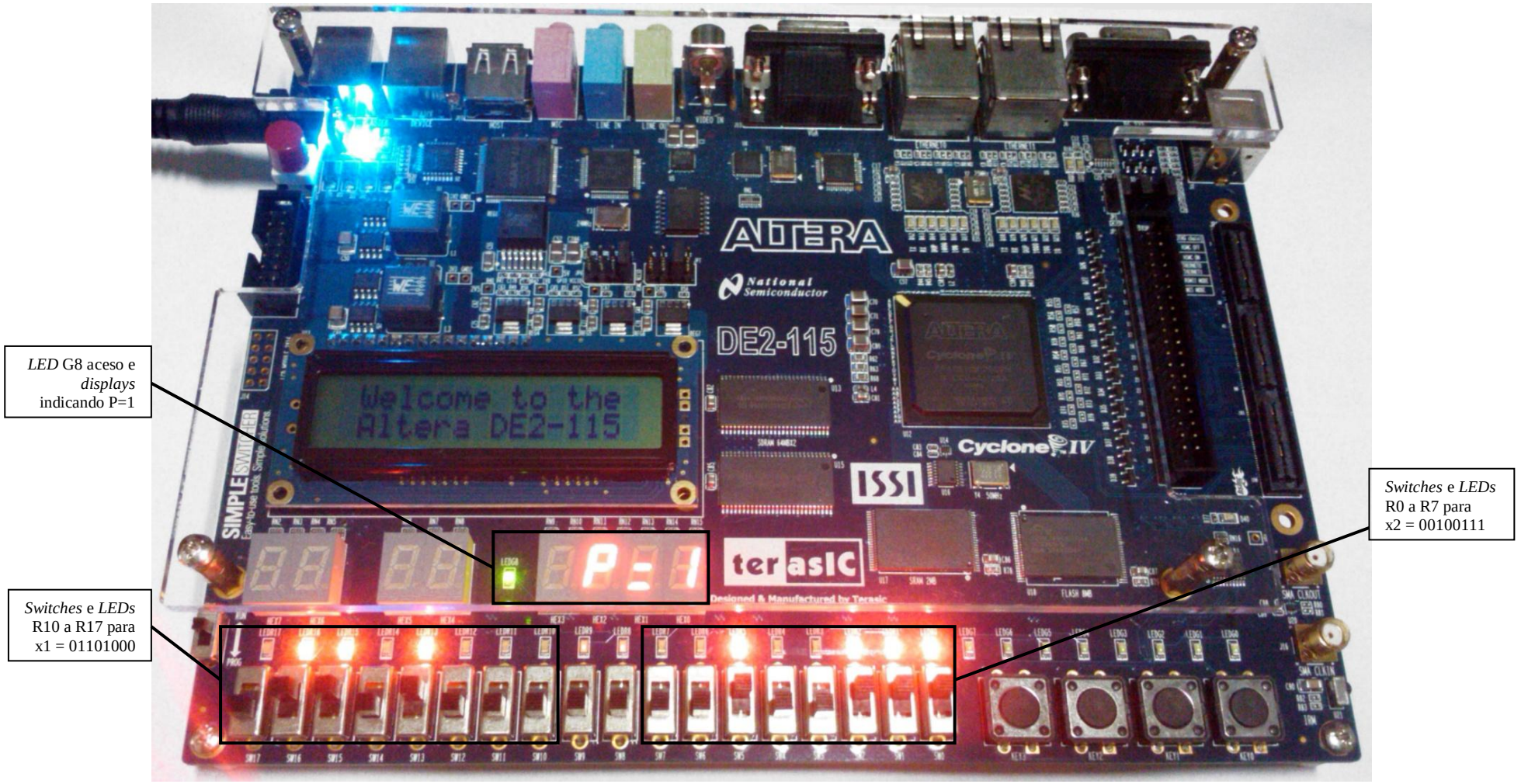


Figura 19 – Circuito da rede *Perceptron* implementado no *kit* DE2-115 (Teste 8 da Tabela 11).

4.2. Variações no número de *bits*

A Tabela 12 relaciona o número de *bits* das entradas e parâmetros de cada uma das variações da rede *Perceptron*, apresentadas na seção 3.5 (pág. 30), com o respectivo número de elementos lógicos exigidos do FPGA para a síntese do circuito equivalente.

Tabela 12 – Variação do Número de *Bits* com Número de Elementos Lógicos Necessários.

Número do Teste	Entrada / Parâmetro	Número de *bits*	Elementos Lógicos
1.0	x_1	8	108
	x_2	8	
	t	8	
	w_1	10	
	w_2	10	
1.1	x_1	6	73
	x_2	6	
	t	6	
	w_1	8	
	w_2	8	
1.2	x_1	10	168
	x_2	10	
	t	10	
	w_1	12	
	w_2	12	
1.3	x_1	12	217
	x_2	12	
	t	12	
	w_1	14	
	w_2	14	
1.4	x_1	14	281
	x_2	14	
	t	14	
	w_1	16	
	w_2	16	
1.5	x_1	10	142
	x_2	10	
	t	10	
	w_1	12	
	w_2	12	
1.6	x_1	12	166
	x_2	12	
	t	12	
	w_1	14	
	w_2	14	

1.7	$\mathbf{x_1}$	14	204
	$\mathbf{x_2}$	14	
	t	14	
	$\mathbf{w_1}$	16	
	$\mathbf{w_2}$	16	

4.3. Discussão dos Resultados

A partir dos resultados exibidos na Tabela 12, foi possível traçar o gráfico exibido na Figura 20. O gráfico explicita as diferentes tendências de crescimento para o complemento dos valores dos parâmetros com *bits* "0" ou *bits* "1". Embora em uma aplicação prática não seja possível escolher o valor dos *bits* a serem utilizados, quaisquer valores de $\mathbf{w_1}$, $\mathbf{w_2}$ e **t** consumirão um número de elementos lógicos compreendido na área entre as duas curvas, para as precisões numéricas testadas.

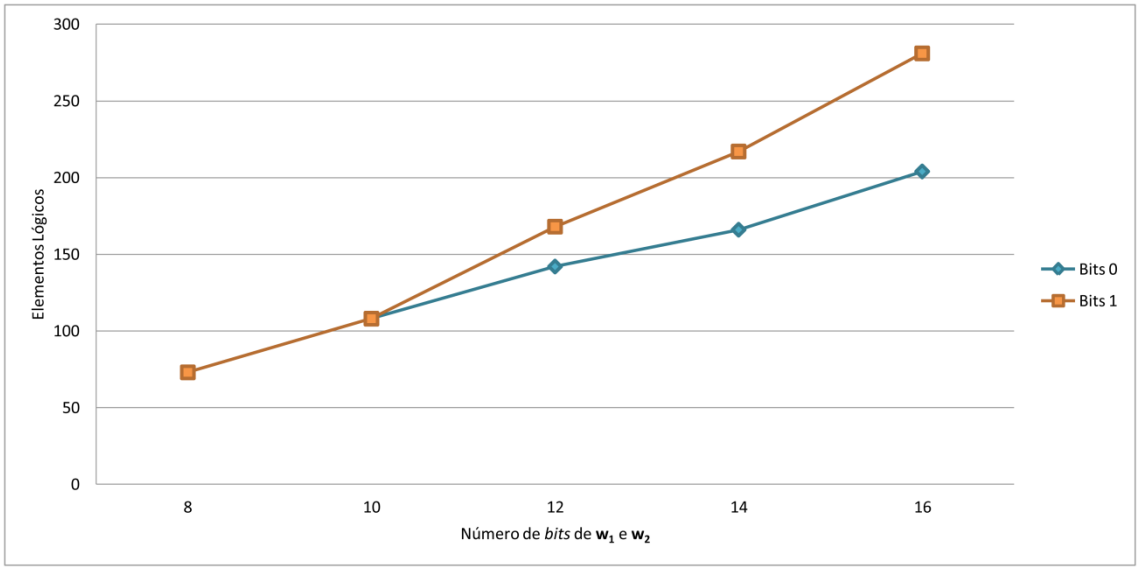


Figura 20 – Relação entre elementos lógicos e números de *bits* (Tabela 12).

Observando-se a tendência de crescimento das linhas exibidas no gráfico é possível, portanto, estimar o número máximo e mínimo de elementos lógicos que serão necessários para sintetizar a rede neural com o uso de precisões numéricas maiores que as apresentadas. O gráfico da Figura 21 exibe a estimativa de crescimento das curvas apresentadas no gráfico da Figura 20 para uma representação de até 24 *bits* dos parâmetros $\mathbf{w_1}$ e $\mathbf{w_2}$.

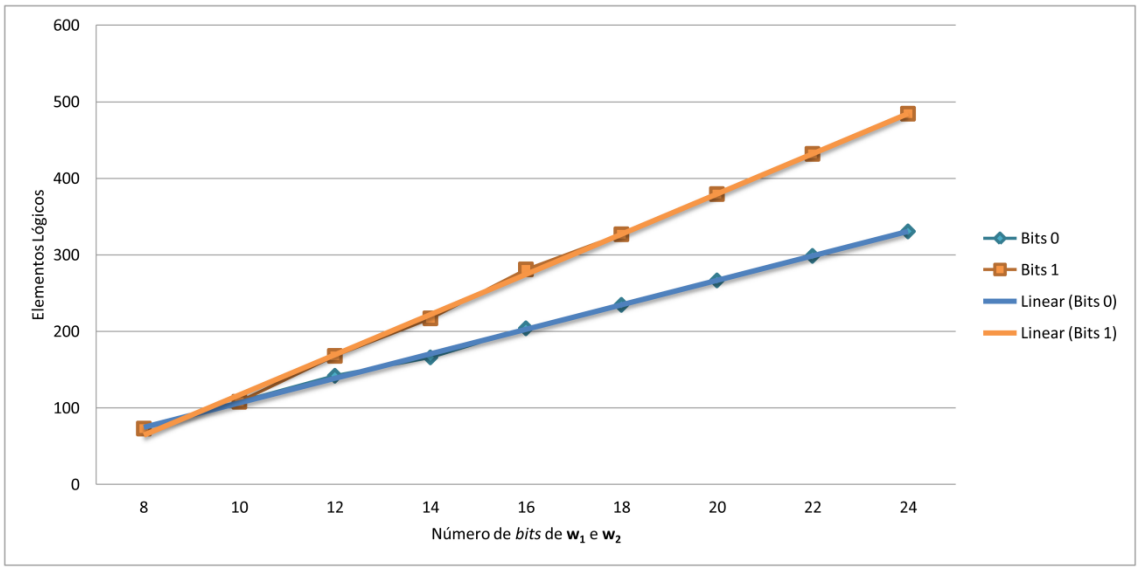


Figura 21 – Estimativa de elementos lógicos necessários até 24 b*its*.

Embora o maior número de elementos lógicos utilizados pela rede sintetizada ainda tenha se mantido abaixo de 1% do total de elementos lógicos disponíveis no FPGA Cyclone IV EP4CE115F29C7 (total de 114.480 elementos lógicos), o fato de que a quantidade necessária deste recurso dobrou do teste 1.0 para o teste 1.3 (Tabela 12), com uma diferença de quatro *bits* na representação numérica, permite observar a importância de uma representação adequada dos dados na síntese de redes neurais em FPGA. Em aplicações com RNAs complexas, uma representação numérica inadequada pode impossibilitar a síntese em dispositivos economicamente viáveis.

5. Conclusão

A Tabela 11 permite observar que a rede Perceptron sintetizada no FPGA apresentou na saída **y** obtida, os mesmos resultados que a saída **y** teórica da rede Perceptron de referência (seção 3.2, pág. 23), quando aplicadas as mesmas entradas. Portanto, o método utilizado permite a síntese de redes neurais em FPGAs, utilizando a linguagem VHDL, com operações de adição e multiplicação em paralelo ao nível de *hardware*, como alternativa às redes neurais executadas por máquinas seqüenciais, que consistem em redes de elementos paralelos processados por *hardwares* que executam as tarefas de adição e multiplicação seqüencialmente.

A discussão dos resultados apresentada na seção 4.3 (pág. 38) permite estimar limites mínimos e máximos para representações numéricas com diferentes precisões e explicita a importância de uma representação adequada dos dados quanto ao requisito de elementos lógicos do FPGA.

A utilização de uma linguagem de descrição de hardware simplificou o processo de desenvolvimento, pois é relacionada ao comportamento que se deseja obter do circuito a ser sintetizado, não exigindo, necessariamente, o conhecimento deste circuito.

A declaração de tipos genéricos permitiu a alteração de valores constantes presentes ao longo da descrição do circuito – para os testes da seção 3.5 (pág. 30) – através da simples mudança do valor declarado no início do arquivo. Além de uma alternativa rápida para substituição de valores, o uso deste recurso evitou que valores que deveriam ser substituídos fossem esquecidos.

A utilização do componente ***decoder*** permitiu o uso de um circuito anteriormente testado e sintetizado na rede neural implementada neste trabalho. O uso deste recurso reduziu simplificou o processo de desenvolvimento, ao permitir que um circuito pré-existente, desenvolvido para outros fins, fosse utilizado no projeto, eliminando a necessidade de projeto de um circuito exclusivo para esta aplicação.

Tal utilização de componentes pré-existentes mostra-se como uma alternativa para a criação de redes de múltiplas camadas, através da reutilização de entidades menores pra formar uma rede neural complexa.

6. Trabalhos Futuros

O trabalho apresentado pode ser estendido das seguintes formas:

- Adição de mais entradas à rede *Perceptron* sintetizada;
- Adequação dos valores de entradas e parâmetros para que sejam representados por números inteiros e comparação com o método apresentado neste trabalho.
- Utilização da rede *Perceptron* sintetizada como um componente para uma rede multicamada;
- Implementação de um processo de treinamento para a rede *Perceptron* sintetizada;
- Síntese de outras topologias de redes em FPGA.
- Implementação de uma RNA em FPGA através da declaração de *look-up tables* com diferentes representações numéricas e comparação com o método apresentado neste trabalho.
- Elaborar um método de entrada simultânea dos *bits* para análise do tempo de processamento para RNAs implementadas em FPGA.

7. Bibliografia

Bélanger N. L. (2007). *Combining DSPs and FPGAs in next-generation multimode wireless handset designs.* Fonte: DSP-FPGA.com: http://dsp-fpga.com/article-id/?2095

Colman D. L. (2006). *Projeto de Redes Neurais Artificiais Aplicadas em Problemas de Automação Industrial.*

Coric S., Latinovic I., Pavasovic A. (2000). A Neural Network FPGA Implementation. *5th seminar on Neural Network Applications in Electrical Engineering* .

da Silva I. N., Spatti D. H., Flauzino R. A. (2010). *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas.*

d'Amore R. (2005). *VHDL, Descrição e síntese de Circuitos Digitais.*

Domingos P. O., Silva F. M., Neto H. C. (2005). An Efficient and Scalable Architecture for Neural Networks with Backpropagation Learning. *International Conference on Field Programmable Logic and Applications.*

Hariprasath S., Prabakar T. N. (2012). FPGA implementation of multilayer feed forward neural network architecture using VHDL. *International Conference on Computing Communication and Applications (ICCCA)* , pp. 1-6.

Haykin S. S. (1999). *Redes Neurais.*

Kovács L. Z. (2006). *Redes Neurais Artificiais: Fundamentos e Aplicações.*

Nieto A., Brea V. M., Vilariño D. L. (2009). *An FPGA-based Topographic Computer for Binary Image Processing.* Fonte: InTech: http://www.intechopen.com/books/image-processing/an-fpga-based-topographic-computer-for-binary-image-processing

Souček B, IRIS Group. (1991). *Neural and Intelligent Systems Integration.*

Terasic Technologies Inc. (s.d.). *Altera DE2-115 Development and Education Board*. Acesso em 05 de 09 de 2012, disponível em terasic.com: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502&PartNo=1

The Institute of Electrical and Electronics Engineers, Inc. (1997). IEEE Standard VHDL Synthesis Packages.

8. Anexos

8.1. Componente *Decoder*

A Figura 22 exibe a descrição do componente ***decoder***.

```
1   Library IEEE;
2   USE ieee.std_logic_1164.all;
3
4   ENTITY decoder IS
5      PORT
6      (
7         INT         : IN  NATURAL RANGE 0 TO 13;
8         DISPLAY     : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
9      );
10  END decoder;
11
12  ARCHITECTURE comportamental  OF decoder IS
13
14  BEGIN
15
16     WITH INT SELECT
17        DISPLAY <= "0000001" WHEN 0,
18                   "1001111" WHEN 1,
19                   "0010010" WHEN 2,
20                   "0000110" WHEN 3,
21                   "1001100" WHEN 4,
22                   "0100100" WHEN 5,
23                   "0100000" WHEN 6,
24                   "0001111" WHEN 7,
25                   "0000000" WHEN 8,
26                   "0000100" WHEN 9,
27                   "0001001" WHEN 10,
28                   "1100000" WHEN 11,
29                   "0110001" WHEN 12,
30                   "1000010" WHEN 13;
31
32  END comportamental;
```

Figura 22 – Descrição do Componente *Decoder*.

8.2. Atribuição de Pinos

A Tabela 13 exibe a atribuição dos pinos do FPGA Cyclone IV EP4CE115F19C7 para as portas utilizadas no circuito da RNA sintetizada, descrita na seção 3.4 (pág. 28).

Tabela 13 – Relação Entre Portas do Circuito Sintetizado e Pinos do FPGA.

Porta	Direção	Pino
disp1[0]	Output	PIN_G18
disp1[1]	Output	PIN_F22
disp1[2]	Output	PIN_E17
disp1[3]	Output	PIN_L26
disp1[4]	Output	PIN_L25
disp1[5]	Output	PIN_J22
disp1[6]	Output	PIN_H22
disp2[0]	Output	PIN_M24
disp2[1]	Output	PIN_Y22
disp2[2]	Output	PIN_W21
disp2[3]	Output	PIN_W22
disp2[4]	Output	PIN_W25
disp2[5]	Output	PIN_U23

disp2[6]	Output	PIN_U24
disp3[0]	Output	PIN_AA25
disp3[1]	Output	PIN_AA26
disp3[2]	Output	PIN_Y25
disp3[3]	Output	PIN_W26
disp3[4]	Output	PIN_Y26
disp3[5]	Output	PIN_W27
disp3[6]	Output	PIN_W28
disp4[0]	Output	PIN_V21
disp4[1]	Output	PIN_U21
disp4[2]	Output	PIN_AB20
disp4[3]	Output	PIN_AA21
disp4[4]	Output	PIN_AD24
disp4[5]	Output	PIN_AF23
disp4[6]	Output	PIN_Y19
LEDx1[7]	Output	PIN_H15
LEDx1[6]	Output	PIN_G16
LEDx1[5]	Output	PIN_G15
LEDx1[4]	Output	PIN_F15
LEDx1[3]	Output	PIN_H17
LEDx1[2]	Output	PIN_J16
LEDx1[1]	Output	PIN_H16
LEDx1[0]	Output	PIN_J15
LEDx2[7]	Output	PIN_H19
LEDx2[6]	Output	PIN_J19
LEDx2[5]	Output	PIN_E18
LEDx2[4]	Output	PIN_F18
LEDx2[3]	Output	PIN_F21
LEDx2[2]	Output	PIN_E19
LEDx2[1]	Output	PIN_F19
LEDx2[0]	Output	PIN_G19
x1[7]	Input	PIN_Y23
x1[6]	Input	PIN_Y24
x1[5]	Input	PIN_AA22
x1[4]	Input	PIN_AA23
x1[3]	Input	PIN_AA24
x1[2]	Input	PIN_AB23
x1[1]	Input	PIN_AB24
x1[0]	Input	PIN_AC24
x2[7]	Input	PIN_AB26
x2[6]	Input	PIN_AD26
x2[5]	Input	PIN_AC26
x2[4]	Input	PIN_AB27
x2[3]	Input	PIN_AD27
x2[2]	Input	PIN_AC27
x2[1]	Input	PIN_AC28
x2[0]	Input	PIN_AB28
y	Output	PIN_F17